X-Pedition™ Security Router

XSR CLI Reference Guide
Version 7.6

P/N 9033842-07

enterasys™
Networks that Know

Notice

Enterasys Networks reserves the right to make changes in specifications and other information contained in this document and its Web site without prior notice. The reader should in all cases consult Enterasys Networks to determine whether any such changes have been made.

The hardware, firmware, or software described in this document is subject to change without notice.

IN NO EVENT SHALL ENTERASYS NETWORKS BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS DOCUMENT, WEB SITE, OR THE INFORMATION CONTAINED IN THEM, EVEN IF ENTERASYS NETWORKS HAS BEEN ADVISED OF, KNEW OF, OR SHOULD HAVE KNOWN OF, THE POSSIBILITY OF SUCH DAMAGES.

Enterasys Networks, Inc.
50 Minuteman Road
Andover, MA 01810

Part Number: 9033842-07 September 2005

Documentation URL: http://www.enterasys.com/support/manuals

Documentacion URL: http://www.enterasys.com/support/manuals

Dokumentation http://www.enterasys.com/support/manuals

Enterasys Networks, Inc.
FIRMWARE LICENSE AGREEMENT

BEFORE OPENING OR UTILIZING THE ENCLOSED PRODUCT, CAREFULLY READ THIS LICENSE AGREEMENT.

This document is an agreement ("Agreement") between the end user ("You") and Enterasys Networks, Inc. on behalf of itself and its Affiliates (as hereinafter defined) ("Enterasys") that sets forth Your rights and obligations with respect to the Enterasys software program/firmware installed on the Enterasys product (including any accompanying documentation, hardware or media) ("Program") in the package and prevails over any additional, conflicting or inconsistent terms and conditions appearing on any purchase order or other document submitted by You. "Affiliate" means any person, partnership, corporation, limited liability company, or other form of enterprise that directly or indirectly through one or more intermediaries, controls, or is controlled by, or is under common control with the party specified. This Agreement constitutes the entire understanding between the parties, and supersedes all prior discussions, representations, understandings or agreements, whether oral or in writing, between the parties with respect to the subject matter of this Agreement. The Program may be contained in firmware, chips or other media.

BY INSTALLING OR OTHERWISE USING THE PROGRAM, YOU REPRESENT THAT YOU ARE AUTHORIZED TO ACCEPT THESE TERMS ON BEHALF OF THE END USER (IF THE END USER IS AN ENTITY ON WHOSE BEHALF YOU ARE AUTHORIZED TO ACT, "YOU" AND "YOUR" SHALL BE DEEMED TO REFER TO SUCH ENTITY) AND THAT YOU AGREE THAT YOU ARE BOUND BY THE TERMS OF THIS AGREEMENT, WHICH INCLUDES, AMONG OTHER PROVISIONS, THE LICENSE, THE DISCLAIMER OF WARRANTY AND THE LIMITATION OF LIABILITY. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT OR ARE NOT AUTHORIZED TO ENTER INTO THIS AGREEMENT, ENTERASYS IS UNWILLING TO LICENSE THE PROGRAM TO YOU AND YOU AGREE TO RETURN THE UNOPENED PRODUCT TO ENTERASYS OR YOUR DEALER, IF ANY, WITHIN TEN (10) DAYS FOLLOWING THE DATE OF RECEIPT FOR A FULL REFUND.

IF YOU HAVE ANY QUESTIONS ABOUT THIS AGREEMENT, CONTACT ENTERASYS NETWORKS, LEGAL DEPARTMENT AT (978) 684-1000.

You and Enterasys agree as follows:

1) **LICENSE.** You have the non-exclusive and non-transferable right to use only the one (1) copy of the Program provided in this package subject to the terms and conditions of this Agreement.

2) **RESTRICTIONS.** Except as otherwise authorized in writing by Enterasys, You may not, nor may You permit any third party to:

(i) Reverse engineer, decompile, disassemble or modify the Program, in whole or in part, including for reasons of error correction or interoperability, except to the extent expressly permitted by applicable law and to the extent the parties shall not be permitted by that applicable law, such rights are expressly excluded. Information necessary to achieve interoperability or correct errors is available from Enterasys upon request and upon payment of Enterasys' applicable fee.

(ii) Incorporate the Program, in whole or in part, in any other product or create derivative works based on the Program, in whole or in part.

(iii) Publish, disclose, copy, reproduce or transmit the Program, in whole or in part.

(iv) Assign, sell, license, sublicense, rent, lease, encumber by way of security interest, pledge or otherwise transfer the Program, in whole or in part.

(v) Remove any copyright, trademark, proprietary rights, disclaimer or warning notice included on or embedded in any part of the Program.

3) **APPLICABLE LAW**. This Agreement shall be interpreted and governed under the laws and in the state and federal courts of the Commonwealth of Massachusetts without regard to its conflicts of laws provisions. You accept the personal jurisdiction and venue of the Commonwealth of Massachusetts courts. None of the 1980 United Nations Convention on Contracts for the International Sale of Goods, the United Nations Convention on the Limitation Period in the International Sale of Goods, and the Uniform Computer Information Transactions Act shall apply to this Agreement.

4) **EXPORT RESTRICTIONS.** You understand that Enterasys and its Affiliates are subject to regulation by agencies of the U.S. Government, including the U.S. Department of Commerce, which prohibit export or diversion of certain technical products to certain countries, unless a license to export the Program is obtained from the U.S. Government or an exception from obtaining such license may be relied upon by the exporting party.

If the Program is exported from the United States pursuant to the License Exception CIV under the U.S. Export Administration Regulations, You agree that You are a civil end user of the Program and agree that You will use the Program for civil end uses only and not for military purposes.

If the Program is exported from the United States pursuant to the License Exception TSR under the U.S. Export Administration Regulations, in addition to the restriction on transfer set forth in Sections 1 or 2 of this Agreement, You agree not to (i) reexport or release the Program, the source code for the Program or technology to a national of a country in Country Groups D:1 or E:2 (Albania, Armenia, Azerbaijan, Belarus, Bulgaria, Cambodia, Cuba, Estonia, Georgia, Iraq, Kazakhstan, Kyrgyzstan, Laos, Latvia, Libya, Lithuania, Moldova, North Korea, the People's Republic of China, Romania, Russia, Rwanda, Tajikistan, Turkmenistan, Ukraine, Uzbekistan, Vietnam, or such other countries as may be designated by the United States Government), (ii) export to Country Groups D:1 or E:2 (as defined herein) the direct product of the Program or the technology, if such foreign produced direct product is subject to national security controls as identified on the U.S. Commerce Control List, or (iii) if the direct product of the technology is a complete plant o r any major component of a plant, export to Country Groups D:1 or E:2 the direct product of the plant or a major component thereof, if such foreign produced direct product is subject to national security controls as identified on the U.S. Commerce Control List or is subject to State Department controls under the U.S. Munitions List.

5) **UNITED STATES GOVERNMENT RESTRICTED RIGHTS.** The enclosed Program (i) was developed solely at private expense; (ii) contains "restricted computer software" submitted with restricted rights in accordance with section 52.227-19 (a) through (d) of the Commercial Computer Software-Restricted Rights Clause and its successors, and (iii) in all respects is proprietary data belonging to Enterasys and/or its suppliers. For Department of Defense units, the Program is considered commercial computer software in accordance with DFARS section 227.7202-3 and its successors, and use, duplication, or disclosure by the Government is subject to restrictions set forth herein.

6) **DISCLAIMER OF WARRANTY.** EXCEPT FOR THOSE WARRANTIES EXPRESSLY PROVIDED TO YOU IN WRITING BY ENTERASYS, ENTERASYS DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT WITH RESPECT TO THE PROGRAM. IF IMPLIED WARRANTIES MAY NOT BE DISCLAIMED BY APPLICABLE LAW, THEN ANY IMPLIED WARRANTIES ARE LIMITED IN DURATION TO THIRTY (30) DAYS AFTER DELIVERY OF THE PROGRAM TO YOU.

7) **LIMITATION OF LIABILITY.** IN NO EVENT SHALL ENTERASYS OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS, PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR RELIANCE DAMAGES, OR OTHER LOSS) ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM, EVEN IF ENTERASYS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS FOREGOING LIMITATION SHALL APPLY REGARDLESS OF THE CAUSE OF ACTION UNDER WHICH DAMAGES ARE SOUGHT.

THE CUMULATIVE LIABILITY OF ENTERASYS TO YOU FOR ALL CLAIMS RELATING TO THE PROGRAM, IN CONTRACT, TORT OR OTHERWISE, SHALL NOT EXCEED THE TOTAL AMOUNT OF FEES PAID TO ENTERASYS BY YOU FOR THE RIGHTS GRANTED HEREIN.

8) **AUDIT RIGHTS.** You hereby acknowledge that the intellectual property rights associated with the Program are of critical value to Enterasys and, accordingly, You hereby agree to maintain complete books, records and accounts showing (i) license fees due and paid, and (ii) the use, copying and deployment of the Program. You also grant to Enterasys and its authorized representatives, upon reasonable notice, the right to audit and examine during Your normal business hours, Your books, records, accounts and hardware devices upon which the Program may be deployed to verify compliance with this Agreement, including the verification of the license fees due and paid Enterasys and the use, copying and deployment of the Program. Enterasys' right of examination shall be exercised reasonably, in good faith and in a manner calculated to not unreasonably interfere with Your business. In the event such audit discovers non-compliance with this Agreement, including copies of the Program made, used or deployed in breach of this Agreement, You shall promptly pay to Enterasys the appropriate license fees. Enterasys reserves the right, to be exercised in its sole discretion and without prior notice, to terminate this license, effective immediately, for failure to comply with this Agreement. Upon any such termination, You shall immediately cease all use of the Program and shall return to Enterasys the Program and all copies of the Program.

9) **OWNERSHIP.** This is a license agreement and not an agreement for sale. You acknowledge and agree that the Program constitutes trade secrets and/or copyrighted material of Enterasys and/or its suppliers. You agree to implement reasonable security measures to protect such trade secrets and copyrighted material. All right, title and interest in and to the Program shall remain with Enterasys and/or its suppliers. All rights not specifically granted to You shall be reserved to Enterasys.

10) **ENFORCEMENT.** You acknowledge and agree that any breach of Sections 2, 4, or 9 of this Agreement by You may cause Enterasys irreparable damage for which recovery of money damages would be inadequate, and that Enterasys may be entitled to seek timely injunctive relief to protect Enterasys' rights under this Agreement in addition to any and all remedies available at law.

11) **ASSIGNMENT**. You may not assign, transfer or sublicense this Agreement or any of Your rights or obligations under this Agreement, except that You may assign this Agreement to any person or entity which acquires substantially all of Your stock or assets. Enterasys may assign this Agreement in its sole discretion. This Agreement shall be binding upon and inure to the benefit of the parties, their legal representatives, permitted transferees, successors and assigns as permitted by this Agreement. Any attempted assignment, transfer or sublicense in violation of the terms of this Agreement shall be void and a breach of this Agreement.

12) **WAIVER.** A waiver by Enterasys of a breach of any of the terms and conditions of this Agreement must be in writing and will not be construed as a waiver of any subsequent breach of such term or condition. Enterasys' failure to enforce a term upon Your breach of such term shall not be construed as a waiver of Your breach or prevent enforcement on any other occasion.

13) **SEVERABILITY.** In the event any provision of this Agreement is found to be invalid, illegal or unenforceable, the validity, legality and enforceability of any of the remaining provisions shall not in any way be affected or impaired thereby, and that provision shall be reformed, construed and enforced to the maximum extent permissible. Any such invalidity, illegality or unenforceability in any jurisdiction shall not invalidate or render illegal or unenforceable such provision in any other jurisdiction.

14) **TERMINATION.** Enterasys may terminate this Agreement immediately upon Your breach of any of the terms and conditions of this Agreement. Upon any such termination, You shall immediately cease all use of the Program and shall return to Enterasys the Program and all copies of the Program.

Contents

Preface

Chapter 1: Network Management

Observing Syntax and Conventions 1-1
Network Management Commands 1-1
General Network Management Commands 1-2
General Show Commands 1-14
snmp-server Commands 1-16
SNMP Show Commands 1-34
SLA Agent Commands 1-37
RTR-mode Commands 1-43
RTR Show Commands 1-45

Chapter 2: Configuring T1/E1 and T3/E3 Subsystems

Observing Syntax and Conventions 2-55
T1/E1 & T3/E3 Commands 2-55
T1/E1 and T3/E3 Clear and Show Commands 2-74
Drop and Insert Commands 2-80

Chapter 3: Configuring the XSR Platform

Observing Syntax and Conventions 3-83
Platform Commands 3-83
Clock Commands 3-84
Crypto Key Commands 3-85
Other Platform Commands 3-86
SNTP Commands 3-91
Platform Clear and Show Commands 3-94
File System Commands 3-107
Bootrom Monitor Mode Commands 3-121

Chapter 4: Configuring Hardware Controllers

Observing Syntax and Conventions 4-83
Hardware Controller Commands 4-83
Hardware Controller Clear and Show Commands 4-92

Chapter 5: Configuring the Internet Protocol

Observing Syntax and Conventions 5-83
IP Commands 5-83
OSPF Commands 5-84
OSPF Debug and Show Commands 5-104
RIP Commands 5-123
RIP Show Commands 5-136
RTP Header Compression Commands 5-137
Triggered on Demand RIP Commands 5-142
Policy-Based Routing Commands 5-145
PBR Clear and Show Commands 5-148

ARP Commands 5-149
Other IP Commands 5-151
IP Clear and Show Commands 5-168
Network Address Translation Commands 5-182
Virtual Router Redundancy Protocol Commands 5-191
VRRP Clear and Show Commands 5-197

Chapter 6: Configuring the Border Gateway Protocol
Observing Syntax and Conventions 6-83
BGP Configuration Commands 6-83
Route Map Commands 6-110
BGP Set Commands 6-114
BGP Clear and Show Commands 6-122
BGP Debug Commands 6-132

Chapter 7: Configuring IP Multicast
Observing Syntax and Conventions 7-83
PIM Commands 7-89
IGMP Clear and Show Commands 7-95

Chapter 8: Configuring the Point-to-Point Protocol
Observing Syntax and Conventions 8-83
PPP Commands 8-83
PPP Debug, Clear and Show Commands 8-97
Multilink PPP Commands 8-108
Multilink Show Commands 8-122

Chapter 9: Configuring Frame Relay
Observing Syntax and Conventions 9-83
Frame Relay Commands 9-83
Frame Relay Map Class Commands 9-95
Frame Relay Clear and Show Commands 9-102

Chapter 10: Configuring the Dialer Interface
Observing Syntax and Conventions 10-83
Dialer Interface Commands 10-83
Dialer Interface Clear and Show Commands 10-90
Dial Backup Commands 10-93
DOD/BOD Commands 10-96
Dialer Watch Commands 10-103

Chapter 11: ISDN BRI and PRI Commands
Observing Syntax and Conventions 11-83
ISDN Commands 11-83
ISDN Debug and Show Commands 11-92

Chapter 12: Configuring Quality of Service
Observing Syntax and Conventions 12-83
QoS Commands 12-83
Policy-Map Commands 12-84
Class-map Commands 12-101

QoS Show Commands 12-105

Chapter 13: Configuring ADSL

Observing Syntax and Conventions 13-83
ADSL Configuration Commands 13-83
CMV Commands 13-83
Other ADSL Commands 13-87
PPP Configuration Commands 13-99
ATM Clear and Show Commands 13-103

Chapter 14: Configuring the VPN

Observing Syntax and Conventions 14-83
VPN Commands 14-83
PKI commands 14-84
CA Identity Mode Commands 14-84
Other Certificate Commands 14-90
IKE Security Protocol Commands 14-94
ISAKMP Protocol Policy Mode Commands 14-95
Remote Peer ISAKMP Protocol Policy Mode Commands 14-99
Remote Peer Show Commands 14-104
IPSec Commands 14-106
IPSec Clear and Show Commands 14-108
Crypto Map Mode Commands 14-110
Crypto Transform Mode Commands 14-115
Crypto Show Commands 14-118
Interface CLI Commands 14-121
Interface VPN Commands 14-122
Tunnel Commands 14-127
Tunnel Clear and Show Commands 14-132
Additional Tunnel Termination Commands 14-134
DF Bit Commands 14-137

Chapter 15: Configuring DHCP

Observing Syntax and Conventions 15-83
DHCP Commands 15-83
ip address dhcp 15-92
DHCP Clear and Show Commands 15-111

Chapter 16: Configuring Security

Observing Syntax and Conventions 16-83
General Security Commands 16-84
Security Clear and Show Commands 16-91
AAA Commands 16-93
AAA Usergroup Commands 16-94
AAA User Commands 16-97
AAA Method Commands 16-101
AAA Per-Interface Commands 16-111
AAA Debug and Show Commands 16-112
Firewall Feature Set Commands 16-115
Firewall Interface Commands 16-129
Firewall Show Commands 16-133

Preface

This guide describes the Command Line Interface (CLI) commands needed to mount, connect, power-up, and maintain an XSR from Enterasys Networks.

This guide is written for administrators who want to configure the XSR or experienced users who are knowledgeable in basic networking principles.

Contents of the Guide

Information in this guide is arranged as follows:

- *Chapter 1, Network Management,* describes fundamental network control commands.
- *Chapter 2, Configuring the T1/E1 & T3/E3 Subsystems*, details commands for T1/E1 and T3/E3 NIM cards.
- *Chapter 3, Configuring the XSR Platform*, describes platform subsystem commands.
- *Chapter 4, Configuring Hardware Controllers*, describes commands to configure the hardware controllers over serial lines.
- *Chapter 5, Configuring the Internet Protocol*, describes IP commands.
- *Chapter 6, Configuring the Border Gateway Protocol,* details BGP commands.
- *Chapter 7 Configuring IP Multicast,* defines XSR commands for Protocol Independent Multicast - Sparse Mode (PIM-SM) and the Internet Group Management Protocol (IGMP).
- *Chapter 8, Configuring the Point-to-Point Protocol*, describes PPP setup.
- *Chapter 9, Configuring Frame Relay,* details commands to configure Frame Relay.
- *Chapter 10, Configuring the Dialer Interface*, describes commands to set up network connections over the Public Switch Telephone Network, provide a backup link over a dial line, and configure BoD/DoD.
- *Chapter 11, ISDN BRI and PRI Commands*, details commands to set up ISDN.
- *Chapter 12, Configuring Quality of Service,* outlines QoS setup commands.
- *Chapter 13, Configuring ADSL,* describes configuration commands for ADSL including CMV, ATM and associated PPP commands.
- *Chapter 14, Configuring the VPN,* details Virtual Private Network setup.
- *Chapter 15, Configuring DHCP,* describes how to set up Dynamic Host Configuration Protocol.
- *Chapter 16, Configuring Security,* describes configuring access lists, and other commands to protect against various network attacks.

Conventions Used in This Guide

The following conventions are used in this guide:

Caution: Contains information essential to avoid damage to the equipment.

Cautela: Contiene información esencial para prevenir dañar el equipo.

Achtung: Verweißt auf wichtige Informationen zum Schutz gegen Beschädigungen.

Note: Calls the reader's attention to any item of information that may be of special importance.

Bold/En negrilla	Text in boldface indicates values you type using the keyboard or select using the mouse (for example, a:\setup). Default settings may also appear in bold. El texto en negrilla indica valores que usted introduce con el teclado o que selecciona con el mouse (por ejemplo, a:\setup). Las configuraciones default pueden también aparecer en en negrilla.
Italics/It áli ca	Text in italics indicates a variable, important new term, or the title of a manual. El texto en itálica indica un valor variable, un importante nuevo término, o el título de un manual.
SMALL CAPS/	Small caps specify the keys to press on the keyboard; a plus sign (+) between keys indicates that you must press the keys simultaneously (for example, CTRL+ALT+DEL). Las mayusculas indican las teclas a oprimir en el teclado; un signo de más (+) entre las teclas indica que usted debe presionar las teclas simultáneamente (por ejemplo, CTRL+ALT+DEL).
Courier font/Tipo de letra Courier	Text in this font denotes a file name or directory. El texto en este tipo de letra denota un nombre de archivo o de directorio.
+	Points to text describing CLI command. Apunta al texto que describe un comando de CLI.
FastEthernet	FastEthernet and GigabitEthernet references are generally interchangeable throughout this guide. Las referencias a los terminos FastEthernet y GigabitEthernet son generalmente intercambiables en el contenido de esta guia.

Getting Help

For additional support related to the XSR, contact Enterasys Networks using one of the following methods:

World Wide Web	**http://www.enterasys.com**
Phone	(603) 332-9400 1-800-872-8440 (toll-free in U.S. and Canada) For the Enterasys Networks Support toll-free number in your country: **http://www.enterasys.com/support/gtac-all.html**
Internet mail	support@enterasys.com To expedite your message, please type **[xsr]** in the subject line.

FTP Login Password	ftp://ftp.enterasys.com
	anonymous
	your Email address
Acquire the latest image and Release Notes	**http://www.enterasys.com/download**
Additional documentation	**http://www.enterasys.com/support/manuals**
Forward comments or suggestions	techwriting@enterasys.com To expedite your message, type **[techwriting]** in the subject line, and include the document Part Number in the Email.

Before contacting Enterasys Networks for technical support, have the following information ready:

- Your Enterasys Networks service contract number
- A description of the failure
- A description of any action(s) already taken to resolve the problem (e.g., rebooting the unit, reconfiguring modules, etc.)
- The serial and revision numbers of any relevant Enterasys Networks products in the network
- A description of your network environment (layout, cable type, etc.)
- Network load and frame size at the time of the problem
- The XSR's history (i.e., have you returned the device before, is this a recurring problem, etc.)
- Any previous Return Material Authorization (RMA) numbers.

1

Network Management

Observing Syntax and Conventions

The CLI Syntax and conventions use the notation described in the following table.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[*x*]	[ ] Square brackets indicate an optional parameter (*italic*)
[*x* | *y* | *z*]	[|] Square brackets with vertical bar indicate a choice of values
{*x* | *y* | *z*}	{ | } Braces with vertical bar indicate a choice of a required value
[*x* {*y* | *z*}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<*xx*>)	*xx* signifies interface type and number, e.g.: `F1`, `S2/1.0`, `D1`, `M57`, `G3. F` indicates a FastEthernet, and `G` a GigabitEthernet interface.
Next Mode entries display the CLI prompt after a command is entered.	
Sub-command headings are displayed in ***red*** text.	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

Network Management Commands

This chapter includes the following subsets of network management commands:

- "General Network Management Commands" on page 1-2
- "General Show Commands" on page 1-14
- "snmp-server Commands" on page 1-16
- "SNMP Show Commands" on page 1-34
- "SLA Agent Commands" on page 1-37
- "RTR-mode Commands" on page 1-43
- "RTR Show Commands" on page 1-45

General Network Management Commands

banner

This command creates a login banner at the XSR's CLI prompt. Text is entered one line at time and should not exceed 80 characters per line. Each successive entry adds a line to the banner, as shown in the example.

Syntax

banner login *bannerLine*

bannerLine	Text to be displayed at login. A maximum of 50 lines can be written per banner. Text must be enclosed in quotes.

Syntax of the "no" Form

Use the *no* form of this command to remove all banners:

```
XSR(config)#no banner login
```

Mode

Global configuration: **XSR(config)#**

Example

The following example configures a login banner:

```
XSR(config)#banner login "Welcome Larry"
XSR(config)#banner login "You're in the office now"
XSR(config)#banner login "Start working!"
```

configure terminal

This command enters configuration mode from Privileged EXEC mode.

Syntax

```
configure terminal
```

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#configure terminal
```

crypto key dsa

This command generates the Digital Signature Algorithm (DSA) type host key pair (private and public) as well as displays the public key. A unique set of host keys are created each time the XSR reboots but we recommend you generate a new pair of host keys when you believe security may be compromised.

The master encryption key is used to encrypt the keys before being saved in the *hostkey.dat* file in Flash. Access to this file is restricted and it cannot be read or copied. All SSH connection requests use the host keys stored in the *hostkey.dat* file unless none have been generated or the content of the file is corrupted. In those circumstances, default keys are used to secure the connection.

Additional host key behavior is described as follows:

- If you have not generated a master encryption key before using SSH, the XSR will prompt you with the **crypto key master generate** command.
- One to three minutes will elapse while host keys are generated by **crypto key dsa**, depending on the device load at the time.
- SSH accepts no new connections during host key generation.
- The command is ignored if stored in the *startup-config* file.
- If the master key is changed, you are not required to generate a new DSA key pair.
- If you remove the master key, the DSA key pair is removed as well (*hostkey.dat* is deleted).

Syntax

crypto key dsa {**generate** | **remove** | **show**}

generate	Produce new key pairs.
remove	Delete old key pair.
show	Display public portion of host key pairs.

Mode

Global configuration: **XSR(config)#**

Example

The following example generates a new pair of keys:

```
XSR(config)#crypto key dsa generate
```

disable

This command exits from Privileged EXEC to EXEC mode.

Syntax

disable

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#disable
```

enable

This command jumps to Privileged EXEC mode.

Syntax

```
enable
```

Mode

EXEC: **XSR>**

Example

```
XSR>enable
```

end

This command terminates configuration mode.

Syntax

```
end
```

Mode

Any configuration

Example

```
XSR(config)#end
```

exit

This command quits the current mode to a higher level. If you are in EXEC mode, it terminates the Telnet, SSH, or Console session.

Syntax

```
exit
```

Mode

All

Example

```
XSR(config)#exit
```

help

This command retrieves help at any Mode.

Syntax

help

Mode

All

Example

XSR#help

ip http port

This command changes the HTTP (Hyper Text Transfer Protocol) port where incoming HTTP (Web) sessions are connecting to.

Syntax

ip http port {*port_number* | default}

port_number	Incoming HTTP server port number from 1024 to 65535.
default	Sets the HTTP port to default.

Note: If you try to set the port-number but it is already in use (Telnet, e.g.) , it will be reset to the default value automatically.

Mode

Global configuration: **XSR(config)#**

Default

Port number: 80

Example

XSR(config)#ip http port 1234

ip http server

This command enables/disables HTTP (Web) service to the router. If the optional parameter is not supplied, the HTTP server will be enabled. Since the HTTP server is *disabled* at boot-up, you must either manually enable it using the CLI or enable it in the **startup-config** file.

Syntax

ip http server [*enable* | *disable*]

enable	Enables HTTP server.
disable	Disables HTTP server.

Syntax of the "no" Form

The *no* form of this command disables the HTTP server:

no ip http server

Mode

Global configuration: **XSR(config)#**

Default

Disable

Examples

```
XSR(config)#ip http server enable
XSR(config)#no ip http server
```

ip ssh server

This command enables/disables Secure Shell (SSH) service to the client. Because the SSH server is enabled at boot-up, you can either manually disable the SSH server using CLI, or disable the SSH server in the *startup-config* file. If the optional parameter is not supplied, the SSH server will be enabled.

Syntax

ip ssh server [*enable* | *disable*]

enable	Enables SSH server.
disable	Disables SSH server.

Syntax of the "no" Form

The *no* form of this command disables the SSH server:

no ip ssh server

Mode

Global configuration: **XSR(config)#**

Defaults

- Enabled

- Port number 22

Example

```
XSR(config)#ip ssh server enable
```

ip telnet port

This command changes the Telnet port where incoming Telnet sessions connect to.

Syntax

ip telnet port {*port_number* | *default*}

port_number	Incoming Telnet server port number from 1024 to 65535.
default	Sets the Telnet port to the default.

Note: If you try to set the port-number but it is already in use (the Web, e.g.) , it will be reset to the default value automatically.

Mode

Global configuration: **XSR(config)#**

Default

Port number: 23

Examples

```
XSR(config)#ip telnet port 5678
```

ip telnet server

This command enables or disables Telnet service to the XSR. If the optional parameter is not supplied, the Telnet server is enabled.

Since the Telnet server is enabled at boot-up, you must either manually disable it using the CLI or disable it in **startup-config**.

Syntax

ip telnet server [*enable* | *disable*]

enable	Enables Telnet service.
disable	Disables Telnet service.

Syntax of the “no” Form

The *no* form of this command disables the Telnet server:

```
no ip telnet server
```

Mode

Global configuration: **XSR(config)#**

Default

Enabled

Examples

```
XSR(config)#ip telnet server enable
XSR(config)#no ip telnet server
```

ping

This network connectivity command, which applies to IP ping only, sends five echo requests with a configurable packet size and source IP address. Ping stops when responses are received or after five requests are sent.

Syntax

ping *dest_addr* [*source_addr*][*size pkt_size*]

dest_addr	Destination address to be pinged.
source_addr	Source address for the ping packet. If not configured, the Router ID is used.
pkt_size	Payload size, ranging from 1 to 65000.

Mode

Privileged EXEC: **XSR#**

Default

Packet size: 72 bytes

Sample Output

This example shows a timed out ping with an unreachable destination:

```
XSR#ping 134.141.235.1
Type escape sequence to abort
Timeout
Timeout
Timeout
Timeout
Timeout
Packets: Sent = 5, Received = 0, Lost = 5
```

The following example shows a successful ping:

```
XSR#ping 134.141.235.165
Type escape sequence to abort
Reply from 192.168.27.165: 20ms
```

```
Reply from 192.168.27.165: 10ms
Reply from 192.168.27.165: 10ms
Reply from 192.168.27.165: 10ms
Reply from 192.168.27.165: 10ms
Packets: Sent = 5, Received = 5, Lost = 0
```

The following example shows the destination lost after three pings:

```
XSR>ping 134.141.235.165
Reply from 134.141.235.165: 20ms
Reply from 134.141.235.165: 10ms
Reply from 134.141.235.165: 10ms
Timeout
Timeout
Packets: Sent = 5, Received = 3, Lost = 2
```

privilege

This command modifies the username privilege level associated with a particular CLI configuration mode. You can also associate a privilege level with another command or group of commands. The modes which can be set include the following:

- **class-map**
- **configure (global)**
- **controller**
- **exec**
- **interface-dialer**
- **interface-dlci**
- **interface-fastEthernet**
- **interface-loopback**
- **interface-serial**
- **map-class-dialer**
- **map-class-frame-relay**
- **policy-map**
- **policy-map-class**
- **router-ospf**
- **router-rip**
- **subinterface**

This command is used in conjunction with the **username** command to set the privilege level for a user. The **show running-config** command displays user information.

Syntax

privilege operationMode {**level** *value* | **reset**} {**command** | **commandgroup**}

privilege	Associates privilege level with a command.
operationMode	Configuration mode associated with privilege level.

value	Privilege level associated with the mode of operation ranging from 0 to 15 (highest).
reset	Resets the privilege level to the default.
command	Command within that mode to set a privilege for.
commandgroup	Set of commands to associate with a privilege. For example, T1 Controller group commands.

Mode

Global configuration: **XSR(config)#**

Defaults

- Privilege level 0: all statistics (**show**) commands with low-level security such as **show version**, **show clock**, etc.
- Privilege levels 1 through 9 - the following EXEC Mode commands are available: **disable, exit, help, isdn, ping, telnet, terminal**, and **traceroute**. Unless explicitly defined, users having privilege levels 1 - 9 have no access to Privileged EXEC commands.
- Privilege levels 10 through 14 - the following Privileged EXEC mode commands are available: **cd, clear, clock, dir, disable, enable, exit, help, isdn, no, ping, pwd, reload, telnet, terminal, traceroute**, and **verify**. Unless explicitly defined, only level 15 users can access Global Mode commands.
- Privilege level 10: all statistics (**show**) commands with higher level security such as **show running-config, show interface**, etc.
- Privilege level 15: other configuration commands such as **configure, copy, delete, rename, write.** Only an *admin* can issue these commands.
- Any user privilege level automatically inherits all privileges granted to lower privilege levels.
- *Admin* privilege level (15) cannot be changed.
- Privilege for special user *admin*: 15
- Only *administrators* can add, delete, or change user rights.
- Only *administrators* can change privilege levels for commands.
- Users can change their own passwords but not their privilege levels.

Examples

This example sets the privilege level for the **username** command in Global mode to level 6:

```
XSR(config)#privilege configure level 6 username
```

This example resets the privilege level for the **username** command in Global mode to the default:

```
XSR(config)#privilege configure reset username
```

This example sets the privilege level for the **neighbor** command in Router RIP mode to level 13:

```
XSR(config)#privilege router-rip level 13 neighbor
```

session-timeout

This command sets the interval for closing a connection when there is no input. If the keyword *console, ssh,* or *Telnet* is used, the timeout becomes the default value for the next session of the specified type, otherwise, the timeout applies to the current session. When the console session times out, it will sit idle and prompt you for your user ID and password again.

Syntax

session-timeout {*timeout* | **console** *timeout* | **ssh** *timeout* | **telnet** *timeout*}

timeout	Timeout current session. Range: 15 - 35,000 seconds.
console	Timeout for console session. Range: 15 - 35,000 seconds.
ssh	Timeout for *all* SSH sessions. Range: 15 - 35,000 seconds
telnet	Timeout for *all* Telnet sessions. Range: 15 - 35,000 seconds.

Mode

Global configuration: **XSR(config)#**

Defaults

- Timeout: 1,800 seconds
- If neither Console, SSH, nor Telnet is specified, the timeout value will be set for the *current* session.

Example

This example sets the current Console timeout session to 15 seconds:

```
XSR(config)#session-timeout console 15
```

terminal

This command changes the terminal screen width and length.

Syntax

terminal {width | length} *size*

width	Width of the terminal screen in lines.
length	Length of the terminal screen in lines.
size	Line range from 0 to 512.

Mode

Privileged EXEC: **XSR#**

Defaults

- Length: 23 lines

- Width: 132 characters
- 0 means no limit

Example

```
XSR#terminal width 40
XSR#terminal length 40
```

traceroute

This command gathers information regarding the route that IP datagrams follow to a specified destination. This implementation of the **traceroute** utility uses UDP as the transport layer. It transmits three probes for each hop between source and destination.

Syntax

traceroute *dest-addr* [*source-addr*]

dest-addr	Network address of the destination.
source-addr	Source address for the ping packet. If this is not set, the Router ID is used.

Mode

EXEC: **XSR>**

Defaults

- Maximum interval to wait for a response: 3 seconds
- Maximum interval to live: 30 seconds
- Packet size: 40 bytes.

Sample Output

```
XSR>traceroute 140.252.13.65 172.15.57.99
traceroute to 140.252.13.65,30 hops max,40 bytes packets
1.  140.252.13.   3520 ms       10 ms      10 ms
2.  140.252.13.   65120ms       120ms      120ms
```

Parameters in the Response

A probe timeout is signaled by an asterisk ” *”.

Abnormal Termination Signs

!P - Protocol Unreachable

!N - Network Unreachable

!H - Host Unreachable

username

This command adds a user, privilege level, password, and encryption type for those accessing the XSR. Assigning privilege levels lets you control which users can manage selective resources. The **username** command can also be used in conjunction with the **privilege** command to associate usernames with particular configuration modes. For example, if configuring T1/E1 requires that a user have a privilege level of 6 or higher, any user with a privilege of 5 or lower would be prohibited from configuring the T1/E1 controller.

Caution: We recommend that you add no more than 3000 users due to a size limit for the the **user.dat** file. Also, we suggest keeping usernames and passwords as short as possible to avoid breaching the 200 Kbyte limit.

Admin/Administrative Users

There is a special level 15 user called *admin* for which you can set a password by specifying admin name as a user. The default password for admin is null (that is, the zero length string "").

Any user with a privilege of 15 is considered an administrator. In at least one of the five permitted Telnet/SSH sessions, an administrative user must be logged. If the first four sessions are in use by regular users, then the fifth session will only allow an administrator to login, otherwise any user can login to the fifth session. If one of the first four sessions has an administrator logged in already, then the fifth session can be any user. This rule is meant to ensure that the administrator can always login.

The **show running-config** command displays user information. By contrast, consult the **aaa client** command which configures a user with AAA security by the XSR authentication database.

Syntax

username *name* [**privilege** *level*] **password** {***cleartext*** | ***secret*** *type*} ***password***

name	User ID.
privilege	Associates a priority level with this user.
level	Priority associated with this user, ranging from 0 to 15 (highest). If the privilege is changed while the XSR is being set, the change occurs immediately.
password	Associates a password with this username.
cleartext	Password will not be encrypted.
secret	Password will be encrypted.
type	*0* indicates the password is expected to be unencrypted, 5 indicates the input password is expected to be encrypted already, so it will not be encrypted again.
password	The password associated with the specified user ID. Users are stored in the **startup-config** file. If you choose a secret password with an optional parameter of 5, then you must provide the password in encrypted form.

Syntax of the "no" Form

The *no* form of this command deletes a user. If no user exists, the command will be ignored. Also, this command will remove the *admin* user provided it is issued by another administrator.

no username *name*

Note: No user can be deleted if you presently logged in as that user and admin or other level 15 users can not be deleted unless at least one such administrator remains configured.

Mode

Global configuration: **`XSR(config)#`**

Defaults

- Username: *admin*
- Password: "" (null or zero length string)
- New user level: 0 unless explicitly set
- Privilege for special user *admin*: 15
- Users with a privilege level of 15 have the same rights as *admin.*
- Only *admins* can add, delete, or change user rights.
- Only *admins* can change privilege levels for commands.
- Users can change their own passwords but not their privilege levels.

Examples

The following example sets *1stUser* privilege to 6 and *2ndUser* to 0:

```
XSR(config)#username 1stUser privilege 6 password cleartext Sox
XSR(config)#username 2ndUser password cleartext Celtic
```

The example below sets the privilege for *larryc* to 15, with an already coded password:

```
XSR(config)#username larryc privilege 15 password secret 5 J&*I8
```

The following example creates user *larryc* with a privilege of 15 and a password that will be encrypted by the XSR:

```
XSR(config)#username larryc privilege 15 password secret 0 nomar
```

General Show Commands

crypto key dsa show

This command displays the encrypted public key, one of the private/public keys generated by the **`crypto key dsa generate`** command. The private key is not displayed.

Syntax

```
crypto key dsa show
```

Mode

Global configuration: **`XSR(config)#`**

Sample Output

The following output displays public key:

```
XSR(config)#crypto key dsa show
---- BEGIN SSH2 PUBLIC KEY ----
Subject: root
Comment: "1024-bit dsa, administrator@Robo1, Mon Mar 03 2003 05:06:16"
AAAAB3NzaC1kc3MAAACBAIgwEkVM26GpC9L+cu9HnXps8S6Qlrhp7mwGudUYDMETdWj53j
u6umHQPwekw0AsTH256mbFedfilcr+W207db+YKunWh59nan/kHGg1iZpwfeaE2kNO4om2
PqXGqdJd7tEI6Ut0cCV7R9roVUDkhmkWWcxaLL5r+YkIV7II6b33AAAAFQCO4IaKlgIhPg
W3oRkNWe3mq9iDrwAAAIBKHSIUIf/KkYd9r5bi7Ec8OHTbkCAcZqwH4gJIh8EryaMWAm7c
zjWtSlLNYhz+q5J2uoPKjct4gqxRv4RLo5yKxsSIcgD6WauvANO7yzQ1CRFBAXL9iZZMEa
AhJQbAE1WVXjD61kBmKvrcR2ZDEnpRaueAaojF4Rslo66Y6pn77gAAAIAKjfSPLGIXe0gF
JqsEIPkrY+0sMwltOV+zd8NPp/NqkIOxg9kZVASQCn/huAv6Sc3WN/WSQU/BpYu2jI8C1S
1S9BEezin8bNE8YWVLwaG1Fx+GOTEugbgflhgMfNHtzaaHEMfmLq80EJ3jRv+zjwaWYPzT
wuo+3CNydBZSwe7fmA==
---- END SSH2 PUBLIC KEY ----
```

show ip http

This command information about the HTTP (Web) session.

Syntax

```
show ip http
```

Mode

Privileged EXEC: **XSR#**

Sample Output

The following is output from the **ip http** command:

```
XSR#show ip http
HTTP Information:
Home page: index.html
HTTP Server: Disabled
HTTP Port: 80
```

show ip telnet

This command information about the Telnet session.

Syntax

```
show ip telnet
```

Mode

Privileged EXEC: **XSR#**

Sample Output

The following is output from the **ip telnet** command:

```
XSR#show ip telnet
TELNET Information:
Telnet Server: Enabled
Telnet Port: 23
Active Telnet Sessions: 1
```

snmp-server Commands

This command set configures the SNMP agent on the XSR. Currently, SNMP v1/v2 and v3 are supported. All commands are invoked in Global configuration mode. If the SNMP server is disabled, executing any SNMP configuration command except for **snmp-server disable** will automatically turn the SNMP server on after it successfully executes. By default, the SNMP server is disabled at boot-up.

All SNMP Global configuration-level commands have a privilege level of 15 and all **show** commands have a level of 10.

The MIBs listed in Table 1-1 can be accessed on the XSR.

Table 1-1 Supported Proprietary and Standard MIB Objects

MIB	Description
ctron-chassis-mib	XSR components and modules MIB.
Enterasys' Download	*ctron-download-mib.txt* (supported via online download only). This is the only MIB with v1/v2c write access.
PPP LCP	RFC-1471. (pppLqrExtnsTable and pppTests not supported)
PPP IP	RFC-1473.
OSPF	RFC-1850. The following traps are supported: *ospfTrapIfStateChange, ospfTrapVirtIfStateChange, ospfTrapNbrStateChange, ospfTrapVirtNbrStateChange, ospfTrapIfConfigError, ospfTrapVirtIfConfigError*
RIPv2	RFC-1724.
BGP	RFC-1657.
Frame Relay DTE	RFC-2115.
ctron-timed-reset-mib	This MIB provides a count down timer and forces a reset after time expires. Using this MIB to reset the XSR performs correctly only if SNMP system shutdown is enabled with the **snmp-server system-shutdown** command (refer to page 26).
Enterasys Configuration Change	This MIB allows management entities to determine if and when configuration changes have occurred. The MIB reports the number of changes and the time and method of the last change in each of three categories: volatile and non-volatile changes, and firmware upgrades.

Table 1-1 Supported Proprietary and Standard MIB Objects (continued)

MIB	Description
Enterasys Configuration Management	This MIB allows an SNMP management entity to upload and download executable images and configuration files to the XSR and identify the active executable image and configuration files. Using this MIB to reset the XSR will succeed only if SNMP system shutdown is enabled with the **`snmp-server system-shutdown`** command (see page 1-27).
Enterasys Syslog Client	The XSR allows read-only access to the Syslog server configuration.
Enterasys SNMP Persistence	This MIB lets SNMP save configuration changes to the *startup-config* file. When reconfiguration occurs via SNMP or the CLI, changes remain volatile until *running-config* is saved to *startup-config*. By setting *etsysSnmp PersistenceSave* to *save (2)*, *running-config* is saved to *startup-config*. The only *etsysSnmpPersistenceMode* supported is *pushButtonSave (2)*.
Enterasys Firewall	This MIB implements SNMP-based Firewall monitoring of the XSR.
Host Resource	RFC-2790. This MIB provides monitoring of CPU load and memory.
Entity MIB V2	RFC-2737. This MIB contains tables for physical and logical entities managed by the SNMP agent.
SNMPv3 MIBs	The SNMPv3 MIBs implemented on the XSR's are: *RFC-3411 Framework, RFC-3412 MPD, RFC-3414 USM, RFC-3415 VACM*
MIB-II	RFC-1213. All objects except the EGP and AT groups. Address Translation (AT) data can be retrieved from *ipNetToMediaTable*.
Evolution of MIB-II Interfaces Group	RFC-1573. *IfStackTable* translated to SMIv1.
IP Tunnel MIB	RFC-2667. *tunnelIfTable* is supported when VPN is enabled.
IP Forward	RFC-2096. *ipCidrRoute* objects.
Enterasys Service Level Reporting	Response Time Reporter for network monitoring.
Notification & Target	RFC-3413.

You can download Enterasys MIBs from the following Web site:

http://www.enterasys.com/support/mibs/

snmp-server community

This command allows a community string to access MIBs in the XSR.

Syntax

snmp-server *community community-string* [**view** *view-name*][***ro*** | ***rw***] [*access-list-num*]

community-string	Community string with SNMP *v1*/*v2c* access.
view-name	Name of the view defining which MIBs are accessible.
ro	Read-only permission.
rw	Read-write permission.
access-list-num	Standard access-list number ranging from 1 to 99.

Notes: You can configure up to 20 read-only and read-write community strings. Community-based write access is available for the *ct-download MIB only*. For write access to other MIBs, use SNMPv3.

Syntax of the "no" Form

The *no* form of this command removes a community string from both read-only and read-write community tables:

no snmp-server community *community-string*

Defaults

- ro
- v1default

Mode

Global configuration: **XSR(config)#**

Example

The following example creates *MyCommunity* for read-write access and applies ACL *#57*:

```
XSR#snmp-server community MyCommunity rw 57
```

snmp-server contact

This command specifies contact information regarding the SNMP server.

Syntax

snmp-server contact *contact-name*

contact-name	String of up to 255 characters. Values with spaces require quotations.

Syntax of the "no" Form

The *no* form of this command offers no contact information:

no snmp-server contact

Mode

Global configuration: **XSR(config)#**

Default

Null string

Example

```
XSR(config)#snmp-server contact LarryCurtis@enterasys.com
XSR(config)#snmp-server contact "Larry Curtis 508 767-2536"
```

snmp-server enable/disable

This command enables or disables the SNMP server. If the server is disabled, using any *snmp* CLI command will turn it back on.

Syntax

snmp-server {*enable* | *disable*}

enable	Enables the SNMP server.
disable	Disables the SNMP server.

Mode

Global configuration: **XSR(config)#**

Default

Disable

snmp-server enable traps

This command enables traps and informs to be sent. SNMPv1 traps and v3 informs are supported, They are sent to the hosts configured with the **snmp-server host** command.

Syntax

snmp-server enable traps [[*snmp* [*authentication*]] *entity* | *frame-relay* | *bgp* | *ospf*]

snmp	Enables all SNMP traps.
authentication	Enables authentication traps *only*.
entity	Enables all entity traps.
frame-relay	Enables all Frame Relay traps.
bgp	Enables all BGP traps.
ospf	Enables all OSPF traps.

Syntax of the “no” Form

The *no* form of this command disables the sending of specified traps:

no snmp-server enable traps [[*snmp* [*authentication*]] *entity* | *frame-relay*]

Mode

Global configuration: **XSR(config)#**

Default

Disabled

Examples

To enable all SNMP traps, enter the following command:

```
XSR(config)#snmp-server enable traps snmp
```

To enable authentication SNMP traps *only*, enter the following command:

```
XSR(config)#snmp-server enable traps snmp authentication
```

snmp-server engineID

This command specifies a value for the SNMP engine on the XSR. Within SNMP v3, users are localized to the device by this Engine ID.

A textual convention for SnmpEngineID is specified by RFC-3411. Using this textual convention, the Engine ID is created with the MAC address and enterprise number for Enterasys. In order to transmit v3 informs, the XSR requires the engineIDs of remote SNMP entities which this command allows you to configure. The command also lets you configure the XSR local engineID.

All engineID settings must be set before adding users to the User Security Model (USM) table since user keys are localized with the engineID.

Caution: If you want to change the engine ID, do so *before* adding SNMP v3 users because you cannot delete a user which is associated with a discarded Engine ID. But you can delete an SNMP user when the Engine ID it is associated with still exists.

Syntax

snmp-server engineID [**local** | **remote** *ip-addr* {**udp-port** *port*}] *engineid-string*

local	The engine-ID is for the local SNMP agent.
remote	The engine-ID is for the remote SNMP agent.
ip-addr	The IP address of the remote host.
port	The UDP port of the remote IP address.
engineid-string	A unique hexadecimal string used to set the local engine ID according to the algorithm defined in RFC-3411. The string must be an even number of up to 54 hex characters.

Syntax of the “no” Form

Use the *no* form of this command to remove the engineID:

no snmp-server engineID [**local** | **remote** *ip-addr* {**udp-port** *port*}] *engineid-string*

Mode

Global configuration: **XSR(config)#**

Example

The following example specifies the Engine ID:

XSR(config)#snmp-server engineID local 00020AF100
results in an engine ID of 0x800015F80500020AF100

snmp-server group

This command configures a new SNMP group to associate SNMP users with views.

Syntax

snmp-server group *group-name* {**v1** | **v2c** | **v3** {**auth | noauth | priv**}} [**read** *readview*] [**write** *writeview*][**access** *access-list*]

group	Defines a User Security Model (USM) group.
group-name	Name of the group.
v1	v1 security model (least secure) used.
v2c	v2 security model (next to least secure) used.
v3	v3 security model (most secure) used.
auth	*authNoPriv* security level used.
noauth	*noAuthNoPriv* security level used.
priv	*authPriv* security level used.
read	Specifies a read view for the group.
readview	The read view name.
write	Specifies a *write* view for the group.
writeview	The write view name.
access	Access-list associated with this group.
access-list	Standard IP access-list allowing access with this group.

Syntax of the “no” Form

Use the *no* form of this command to remove a specified SNMP group:

no snmp-server group *group-name* {**v1** | **v2c** | **v3**}{**auth** | **noauth** | **priv**}}

Mode

Global configuration: **XSR(config)#**

Example

This example specifies the *v3auth* SNMP group with *auth security,* the *v3* view for read and write access, and is matched with an ACL written earlier:

XSR(config)#snmp-server group v3auth v3 auth read v3view write v3view access 88

snmp-server host

This command specifies host parameters of the SNMP server; it adds a new management station to send traps to. If the address already exists, the command will update the server's configuration which is stored in the *snmpTarget* MIB defined by RFC-2573.

Syntax

snmp-server host *ip-addr* {**traps** | **informs version** {**2c** | **3** [{**auth** | **noauth** | **priv**}]] *community-stringOrUser* [**udp-port** *port*][*notification-type*]

ip-addr	IP address of the target recipient.
traps	Sends SNMP traps to this host.
informs	Sends Inform notifications.
version	The security model used.
2c	Version 2c security model used. This allows the transmission of *informs* and *counter64* values.
3	Version 3 security model (USM) used.
auth	Authentication *without* encryption.
noauth	No authentication or encryption.
priv	Authentication *with* encryption.
community-stringOrUser	Password-like community string to be used with for versions 1 and 2c. User name when using version 3 security model.
udp-port	Specifies the UDP port of the host to use.
port	The UDP port number of the host.
notification-type	The type of trap to be sent including *BGP, entity, frame-relay, ospf*, and *snmp* traps.

Note: You can configure up to 20 hosts.

Syntax of the "no" Form

The *no* form removes the specified *host* from the list of hosts that the XSR sent traps to:

no snmp-server host *host* *ip-addr*

Mode

Global configuration: **XSR(config)#**

Defaults

- Trap-type: SNMP, entity, frame-relay
- UDP port: 162

Example

The following examples illustrate an SNMP host with trap *on* and *off*:

```
XSR(config)#snmp-server host 192.168.1.10 traps trapsOn
XSR(config)#no snmp-server host 192.168.2.11
```

Sample Output

The following are three sample outputs from the command:

```
Notification host: 192.168.2.10 udp-port: 162   type: inform
user: v3user    security model: v3 priv

Notification host: 192.168.10.2 udp-port: 162   type: trap
user: public    security model: v1

Notification host: 192.168.1.5  udp-port: 162   type: trap
user: testuser  security model: v3 noauth
```

snmp-server informs

This command specifies inform request options.

Syntax

snmp-server informs [**retries** retries] [**timeout** *seconds*] [**pending** *pending*]

Syntax of the "no" Form

The *no* form of this command returns settings to their defaults:

no snmp-server informs [**retries** retries][**timeout** *timeout*] [**pending** *pending*]

retries	Maximum attempts to resend an inform request. Range: 0 -10.
timeout	Interval to wait for an acknowledgement before resending. Range: 1 - 10 seconds.
pending	Peak number of informs waiting for acknowledgments at any one time, ranging from 1 to 100. When the peak is reached, older pending informs are discarded.

Mode

Global configuration: **XSR(config)#**

Defaults

- Retries: 3
- Timeout: 15 seconds
- Pending: 25 informs

Example

This example shows an inform with *1* retry, a *5*-second timeout and a *10* pending value:

```
XSR(config)#snmp-server informs retries 1 timeout 5 pending 10
```

snmp-server location

This command specifies the location of the SNMP server.

Syntax

snmp-server location *location-string*

location-string	Site where the SNMP server is located.

Syntax of the “no” Form

The *no* form of this command deletes a location for the SNMP server:

```
no snmp-server location
```

Mode

Global configuration: **XSR(config)#**

Default

Null string

Example

The following example describes the SNMP server location. Note the quotation marks:

```
XSR(config)#snmp-server location “Beacon Street Branch”
```

snmp-server max-traps-per-window

This command specifies the number of traps allowed in the time window.

Syntax

snmp-server max-traps-per-window *max-traps*

max-traps	Sum of traps permitted, ranging from 0 to 999,999,999.

Syntax of the “no” Form

The *no* form of this command sets the minimum period between successive traps to the default:

```
no snmp-server max-traps-per-window
```

Mode

Global configuration: **XSR(config)#**

Default

0 traps (unlimited)

Example

The following example sets the traps permitted to 1000:

```
XSR(config)#snmp-server max-traps-per-window 1000
```

snmp-server min-trap-spacing

This command sets the interval between successive SNMP traps. Trap spacing is only guaranteed to occur at least every spacing - it might occur more often. The command implementation can exhibit a jitter of +0 to +200 milliseconds and is linked to the XSR's fast timer tick interval.

Syntax

snmp-server min-trap-spacing *spacing*

spacing	Minimum interval between successive traps, ranging from 0 to 3,600,000 milliseconds. Zero (0) indicates traps are sent successively, without delay.

Syntax of the "no" Form

The *no* formsets the minimum interval between successive traps to the default value:

```
no snmp-server min-trap-spacing
```

Mode

Global configuration: **XSR(config)#**

Default

200 milliseconds

Example

The following example limits the minimum trap interval to *1* minute:

```
XSR#snmp-server min-trap-spacing 60000
```

snmp-server packetsize

This command sets the maximum allowable incoming and outgoing packet size in bytes. Packets larger than this value are dropped.

Syntax

snmp-server packetsize *size*

size	Peak packet size allowed, ranging from 484 to 8,192 bytes.

Syntax of the “no” Form

The *no* form sets the maximum allowed incoming and outgoing *packetsize* to the default:

```
no snmp-server packetsize
```

Mode

Global configuration: **XSR(config)#**

Default

1,500 bytes

Example

The following example specifies the peak packet size as *1000* bytes:

```
XSR#snmp-server packetsize 1000
```

snmp-server queue-length

This command sets the retransmission queue length. Traps which have no route to the host are put into the retransmission queue for resending later.

Syntax

snmp-server queue-length *length*

length	Trap queue length ranging from 1 to 1000.

Syntax of the “no” Form

The *no* command resets the retransmission queue *length* to the default:

```
no snmp-server queue-length
```

Mode

Global configuration: **XSR(config)#**

Default

10

Example

The following example sets the retransmission queue length to *50*:

```
XSR#snmp-server queue-length 50
```

snmp-server set entityMIB

This command specifies physical alias and asset IDs for the entity MIB.

Syntax

snmp-server set entityMIB {**entPhysicalAlias** | **entPhysicalAssetID**} **host** <*string*>

entPhysicalAlias	An alias name for the physical entity.
entPhysicalAssetID	A user-assigned asset tracking identifier for the physical entity.
string	Text for the *alias* or *ID* not to exceed 32 characters.

Syntax of the "no" Form

The *no* command sets the *PhysicalAlias* or *PhysicalAssetID* in the Entity MIB as an empty string:

no snmp-server set entityMIB {**entPhysicalAlias** | **entPhysicalAssetID**} **host**

Mode

Global configuration: **XSR(config)#**

Example

The following example provides an alias for the host:

```
XSR(config)#snmp-server set entityMIB entPhysicalAlias host aliasSalesServer
```

snmp-server system-shutdown

This command allows the SNMP server to reboot the XSR (usually after a software download).

Syntax

snmp-server system-shutdown

Syntax of the "no" Form

The *no* command disallows the SNMP server from rebooting the XSR:

no snmp-server system-shutdown

Mode

Global configuration: **XSR(config)#**

Default

Enabled

Example

The following example permits the SNMP server to reboot the XSR:

```
XSR(config)#snmp-server system-shutdown
```

snmp-server tftp-server-list

This command specifies an Access Control List (ACL) to limit TFTP servers' access during SNMP downloads.

Syntax

snmp-server tftp-server-list *access-list-num*

access-list-num	Standard ACL ranging from 1 to 99.

Syntax of the "no" Form

The *no* form removes any ACL limiting other TFTP servers' access during SNMP downloads:

no snmp-server tftp-server-list

Mode

Global configuration: **XSR(config)#**

Example

The following example limits TFTP servers to ACL *#57*:

```
XSR#snmp-server tftp-server-list 57
```

snmp-server trap-source

This command sets the interface serving as the source for all traps and informs. Use the address of the interface from which the trap/inform goes out as the source address for the trap/inform.

Syntax

snmp-server trap-source {*interface*}

interface	A supported interface such as FastEthernet 1.

Note: If the interface does not have an IP address or if the interface is deleted afterwards, it will use the address of the interface from which the trap/inform goes out as the source address for the trap/inform.

Syntax of the "no" Form

The *no* form of this command removes the configured trap interface:

no snmp-server trap-source

Example

This example specifies *GigabitEthernet interface 2* as the trap source:

```
XSR#snmp-server trap-source g2
```

snmp-server trap-timeout

This command specifies the interval traps in the retransmission queue are retried if no route exists to the host that SNMP traps will to be sent to.

Syntax

snmp-server trap-timeout *timeout*

timeout	Retry interval ranging from 1 to 9,999 seconds.

Syntax of the "no" Form

The *no* form of this command sets the *trap-timeout* to the default value:

no snmp-server trap-timeout

Mode

Global configuration: **XSR(config)#**

Default

30 seconds

snmp-server user

This command configures local or remote users in an SNMP group with security models, authentication, passwords, privacy settings, and ACLs, and adding users to the USM user table.

Note: Be aware that the engineID of the remote SNMP entity must be configured before you add a user since passwords are hashed with the engineID to create a localized key.

Syntax

snmp-server user *username* [*groupname* **remote** *ip-address* [**udp-port** *port*]{**v1** | **v2c** | **v3** [**encrypted**][**auth** {**md5** | **sha**} *auth-password* [**priv des56** *priv-password*]]}[**access** *access-list*]

username	Name of the user.
groupname	Name of the group to which the user belongs.
remote	A remote SNMP entity.
ip-address	IP address of the remote SNMP entity.
udp-port	UDP port of the remote SNMP entity.
port	UDP port number of the remote SNMP entity. Default: 162.
v1	v1 security model (least secure) used.
v2c	v2c security model (next to least secure) used.
v3	v3 security model (most secure) used.
encrypted	Specifies passwords as MD5 or SHA digests.
auth	Authentication parameters for the user.
md5	HMAC MD5 algorithm used for authentication.

sha	HMAC SHA algorithm used for authentication.
auth-password	The user's authentication password. At least 8 characters is required.
priv	Specifies the privacy setting.
des56	CBC-DES privacy encryption algorithm.
priv-password	Privacy password for the user. A minimum of 8 characters is required.
access	Specifies an access-list associated to this user.
access-list	Standard IP access-list allowing access to this user.

Syntax of the "no" Form

Use the *no* form of this command to remove a user:

no snmp-server user *username groupname* {**v1** | **v2c** | **v3**}

Mode

Global configuration: **XSR(config)#**

Example

The example below configures *ljc* of the *v3authgrp* SNMP group with strong *v3* level security, *MD5* authentication, and the password *acorntree*:

```
XSR(config)#snmp-server user ljc v3 auth v3authgrp md5 acorntree
```

snmp-server view

This command creates or updates a view entry. The XSR provides one default view which is used for all community commands which do not specify a view parameter. The *v1default* view includes the *internet* tree and excludes *snmpUsmMIB* and *snmpVacmMIB*. You can remove this view with the **no snmp-server v1default** command.

Syntax

snmp-server view *view-name* {*oid-tree* | *treeEntryName*} {**included** | **excluded**}

view-name	Label for the view record that you update/create.
oid-tree	Object identifier of the subtree to be included/excluded from the view. This parameter can be either a numeric OID or a well-known MIB name listed in Table 1-2 on page 1-31, or a MIB name followed by a numeric OID (i.e., system.6 for *sysLocation*). Names are case-sensitive.
treeEntryName	Name of the sub-tree equivalent to the object OID tree.
included	This view includes the specified OID tree.
excluded	This view excludes the specified OID tree.

Syntax of the "no" Form

Use the *no* form of this command to remove a view entry:

no snmp-server view *view-name*

Mode

Global configuration: **`XSR(config)#`**

Examples

The following example creates a view of all objects on the XSR:

```
XSR(config)#snmp-server view v3view internet included
```

The following example creates a view of all objects in the MIB-II subtree:

```
XSR(config)#snmp-server view mib2 mib-2 included
```

The following example creates a view for TCP:

```
XSR(config)#snmp-server view TCPview tcp included
```

The following example creates a view of all objects in the MIB-II subtree excluding 1.3.6.1:

```
XSR(config)#snmp-server view MIBIIview 1.3.6.1 excluded
```

The following example removes a view of MIN-II subtree 1.3.6.1:

```
XSR(config)#no snmp-server view 1.3.6.1
```

The following example creates a view of all objects in private Enterasys and Cabletron MIBs except for the *etsysConfigurationChange* MIB:

```
XSR(config)#snmp-server view Enterasys private included
XSR(config)#snmp-server view Enterasys etsysConfigurationChangeMIB excluded
```

Sample Output

The following is sample output from the command:

```
XSR#show snmp view
viewname:    Enterasys
          included:
             private
          excluded:
             etsysConfigurationChangeMIB
```

Table 1-2 MIB Names for SNMP View Commands

SNMP Term	SNMP Numerical ID
org	1.3
dod	1.3.6
internet	1.3.6.1
mgmt	1.3.6.1.2
private	1.3.6.1.4
snmpV2	1.3.6.1.6
mib-2	1.3.6.1.2.1
system	1.3.6.1.2.1.1
interfaces	1.3.6.1.2.1.2
ifEntry	1.3.6.1.2.1.2.2.1

Table 1-2 MIB Names for SNMP View Commands (continued)

SNMP Term	SNMP Numerical ID
at	1.3.6.1.2.1.3
atEntry	1.3.6.1.2.1.3.1.1
ip	1.3.6.1.2.1.4
ipAddrEntry	1.3.6.1.2.1.4.20.1
ipRouteEntry	1.3.6.1.2.1.4.21.1
ipNetToMediaEntry	1.3.6.1.2.1.4.22.1
icmp	1.3.6.1.2.1.5
tcp	1.3.6.1.2.1.6
tcpConnEntry	1.3.6.1.2.1.6.13.1
udp	1.3.6.1.2.1.7
udpEntry	1.3.6.1.2.1.7.5.1
egp	1.3.6.1.2.1.8
transmission	1.3.6.1.2.1.10
pppLcp	1.3.6.1.2.1.10.23.1
pppIp	1.3.6.1.2.1.10.23.3
frameRelayDTE	1.3.6.1.2.1.10.33
tunnelMIB	1.3.6.1.2.1.10.131
snmp	1.3.6.1.2.1.11
ospf	1.3.6.1.2.1.14
bgp	1.3.6.1.2.1.15
rip2	1.3.6.1.2.1.23
ifMIB	1.3.6.1.2.1.31
entityMIB	1.3.6.1.2.1.47
cabletron	1.3.6.1.4.1.52
chassis	1.3.6.1.4.1.52.4.1.1.2
ctTimedResetMIB	1.3.6.1.4.1.52.4.1.1.5.2
ctDownload	1.3.6.1.4.1.52.4.1.5.8
enterasys	1.3.6.1.4.1.5624
etsysConfigurationChangeMIB	1.3.6.1.4.1.5624.1.2.12
etsysSyslogClientMIB	1.3.6.1.4.1.5624.1.2.14
etsysSnmpPersistenceMIB	1.3.6.1.4.1.5624.1.2.24
etsysFirewallMIB	1.3.6.1.4.1.5624.1.2.37
etsysServiceLevelReportingMIB	1.3.6.1.4.1.5624.1.2.39
snmpFrameworkMIB	1.3.6.1.6.3.10

Table 1-2 MIB Names for SNMP View Commands (continued)

SNMP Term	SNMP Numerical ID
snmpMPDMIB	1.3.6.1.6.3.11
snmpUsmMIB	1.3.6.1.6.3.15
snmpVacmMIB	1.3.6.1.6.3.16
snmpEngine	1.3.6.1.6.3.10.2.1
snmpMPDStats	1.3.6.1.6.3.11.2.1
usmStats	1.3.6.1.6.3.15.1.1
usmUser	1.3.6.1.6.3.15.1.2
usmUserTable	1.3.6.1.6.3.15.1.2.2
vacmContextTable	1.3.6.1.6.3.16.1.1
vacmSecurityToGroupTable	1.3.6.1.6.3.16.1.2
vacmAccessTable	1.3.6.1.6.3.16.1.4
vacmMIBViews	1.3.6.1.6.3.16.1.5
vacmViewTreeFamilyTable	1.3.6.1.6.3.16.1.5.2

snmp-server window-time

This command specifies the length, in seconds, of the moving window used to count the number of traps sent.

Syntax

snmp-server window-time *time*

time	Time window interval, ranging from 1 to 3,600 seconds.

Syntax of the “no” Form

The *no* form of this command sets the length of the moving window used to count the number of traps sent in recently to default:

no snmp-server window-time

Mode

Global configuration: **XSR(config)#**

Default

10 seconds

Example

The following example sets the moving window interval to ten minutes:

```
XSR(config)#snmp-server window-time 600
```

SNMP Show Commands

show snmp

This command information about the SNMP server.

Syntax

show snmp [*location*]

location	The site of the SNMP server.

Mode

Privileged EXEC: **XSR#**

Sample Output

The following is sample output from the command:

```
XSRtop(config)#show snmp
Chassis serial#: 0000019876543210
In counters:
            0 SNMP packets in
           0 Bad SNMP version errors
           0 Unknown community names
           0 Illegal operations for name supplied
           0 Encoding errors
           0 Packets too big
           0 No such names
           0 Bad values
           0 Read-onlys
           0 General Errors
           0 Requested variables
           0 Altered variables
           0 Get requests
           0 Get-Next requests
           0 Set requests
           0 Get responses
           0 Traps
Out counters:
            0 SNMP packets out
           0 Packets too big
           0 No such names
           0 Bad values
           0 General errors
           0 Get requests
           0 Get-Next requests
           0 Set requests
           0 Get responses
           0 Traps
```

```
0 Silent drops
0 Proxy drops
```

The example below shows output with the location option entered:

```
XSR#show snmp location
Haverhill Mass.
```

show snmp engineID

This command displays the identification of the local SNMP engine.

Syntax

```
show snmp engineID
```

Mode

```
Privileged EXEC: XSR#
```

Sample Output

The following is sample output from the command:

```
XSR#show snmp engineID
Local SNMP engineID: 800015F8030001F423E691
IP-addr             Port      Rewrite Engine ID
10.10.1.48          162       800009041234
```

show snmp group

This command displays the names of groups on the XSR with their security model and views.

Syntax

```
show snmp group
```

Mode

```
Privileged EXEC: XSR#
```

Sample Output

The following sample output displays one group, *nm*, which was configured with a few views attached to it:

```
XSR#show snmp group
grouname: nm                  security model: v1
readview: tcpView             wirteview: tcpView
notifyview: <no notifyview specified>

grouname: nm                  security model: v2c
readview: v1default           wirteview: <no writeview specified>
notifyview: <no notifyview specified>
```

```
grouname: nm                    security model: v3 auth
readview: v1default             wirteview: nmMIBIIview
notifyview: nmMIBIIview
```

The following is sample output from the command:

```
XSR#show snmp group
groupname: v3RWGroup            security model: v3
readview: v3view                writeView: v3view
notifyview: <no notifyview specified>

groupname: v3ROGroup            security model: v3
readview: v3view                writeView: nmMIBIIview
notifyview: <no notifyview specified>
```

show snmp host

This command displays information from the SNMP Host table.

Syntax

```
show snmp host
```

Sample Output

The following is sample output from the command:

```
Notification host: 192.168.2.10 udp-port: 162   type: inform
user: v3user    security model: v3 priv

Notification host: 192.168.10.2 udp-port: 162   type: trap
user: public    security model: v1

Notification host: 192.168.1.5  udp-port: 162   type: trap
user: testuser  security model: v3 noauth
```

show snmp user

This command displays information on each SNMP username in the Username table.

Syntax

```
show snmp user
```

Mode

```
Privileged EXEC: XSR#
```

Sample Output

The following is sample output from the command:

```
XSR#show snmp user
```

```
User name: authprivUser              group: v3RWGroup
Engine ID: 800015f8030001f423e691
storage-type: nonvolatile            active
```

Parameter Description

storage-type	Indicates whether the settings have been saved to persistent memory (non-volatile) or will be lost if the device is reset (volatile).

show snmp view

This command displays information on each SNMP view in the group username table.

Syntax

```
show snmp view
```

Mode

```
Privileged EXEC: XSR#
```

Sample Output

The following is sample output from the command:

```
XSR#show snmp view
viewname:           v3view
          included:
                    internet
          excluded:
viewname:           v1default
          included:
                    internet
          excluded:
                    snmpUsmMIB
                    snmpVacmMIB
viewname:           MIBIIview
          included:
                    1.3.6.1
          excluded:
```

SLA Agent Commands

aggregate period

This command specifies the period between two aggregate measurement action intervals by the Response Time Reporter (RTR).

Syntax

aggregate-period *period*

period	Interval between aggregate measurement, ranging from 10 to 60800 seconds.

Syntax of the "no" Form

The *no* form of this command returns to the default value:

aggregate-period *period*

Mode

RTR Echo configuration: **XSR(config-rtr-echo-*xx*)#**

Default

600 seconds

Example

The following example sets a one-minute aggregate period:

```
XSR(config-rtr-echo-1)#aggregate-period 60
```

buckets-of-history-kept

This command specifies how many history entries will be maintained by the Response Time Reporter (RTR).

Syntax

buckets-of-history-kept *size*

size	Number of history records retained. Range: 1 to 60.

Syntax of the "no" Form

The *no* form of this command returns to the default value:

```
no buckets-of-history-kept
```

Mode

RTR Echo configuration: **XSR(config-rtr-echo-*xx*)#**

Default

- Size: 10 records
- The result is wrapped when the history is full.

Example

This example sets the buckets-of-history value to *5* records:

```
XSR(config-rtr-echo-1)#buckets-of-history-kept 5
```

frequency

This command specifies how frequently to send a Response Time Reporter (RTR) probe. The value you configure for frequency must be larger than your configured timeout value so that a user cannot have a frequency of 1 second and a timeout of 1001 milliseconds.

Syntax

frequency {*frequency-interval*}

frequency-interval	How often to send a probe, ranging from 1 to 604,800 seconds.

Syntax of the "no" Form

The *no* form of this command returns to the default value:

```
no frequency
```

Mode

RTR Echo configuration: **XSR(config-rtr-echo-xx)#**

Default

Frequency: 60 seconds

Example

The following example sets the RTR frequency to *2* seconds:

```
XSR(config-rtr-echo-57)#frequency 2
```

map

This command associates a Response Time Reporter (RTR) with a map - an administratively assigned name.

Syntax

map {*map-name*}

map-name	Network management map to which the RTR belongs.

Syntax of the "no" Form

The *no* form of this command returns to the default value:

```
no map
```

Mode

RTR Echo configuration: **XSR(config-rtr-echo-*xx*)#**

Example

The following example creates an RTR map:

```
XSR(config-rtr-echo-57)#map "network in Peoria"
```

owner

This command binds a Response Time Reporter (RTR) owner (administrator) to a measurement entry.

Note: Because the Enterasys service level reporting MIB requires an owner to be created *before* an entry, an owner must be added first.

Syntax

owner {*owner-name*}

owner-name	Owner's name.

Syntax of the “no” Form

The *no* form of this command removes any configured owner:

```
no owner
```

Mode

RTR Echo configuration: **XSR(config-rtr-echo-*xx*)#**

Example

The following example specifies the RTR owner:

```
XSR(config-rtr-echo-57)#owner operator1
```

request-data-size

This command specifies the Response Time Reporter (RTR) payload size.

Syntax

request-data-size {*payload-size*}

payload-size	Requested payload size, ranging from 12 to 16384 bytes.

Syntax of the “no” Form

The *no* form of this command returns to the default value:

```
no request-data-size
```

Mode

RTR Echo configuration: **XSR(config-rtr-echo-*xx*)#**

Default

Payload size: 12 bytes

Example

The following example limits the RTR payload size to *32* bytes:

```
XSR(config-rtr-echo-57)#request-data-size 32
```

tag

This command specifies an identifier (name) for this Response Time Reporter (RTR) measurement.

Syntax

tag {*name-tag*}

name-tag	Name assigned to this measurement.

Syntax of the “no” Form

The *no* form of this command removes any configured tag:

```
no tag
```

Mode

RTR Echo configuration: **XSR(config-rtr-echo-*xx*)#**

Example

The following example specifies the RTR name:

```
XSR(config-rtr-echo-57)#tag "one-way packet loss"
```

timeout

This command specifies a timeout for the Response Time Reporter (RTR). Be aware that the timeout value must be smaller than the frequency value. So, a user cannot have a frequency of 1 second and a timeout of 1001 milliseconds.

Syntax

timeout {*timeout-value*}

timeout-value	Timeout, ranging from 1 to 604800000 milliseconds.

Syntax of the "no" Form

The *no* form of this command returns to the default value:

```
no timeout
```

Mode

RTR Echo configuration: **XSR(config-rtr-echo-*xx*)#**

Default

5000 milliseconds

Example

The following example resets the RTR timeout to *500* milliseconds:

```
XSR(config-rtr-echo-57)#timeout 500
```

type

This command specifies the type of Response Time Reporter (RTR) measurement to be performed - ICMP Echo - as well as the destination and source host IP addresses.

Syntax

type {echo} **protocol** {**ipIcmpEcho**} *dst* [**source-ipaddr** ***src***]

dst	IP address of the destination host.
src	IP address used as the source.

Mode

RTR configuration: **XSR(config-rtr-*xx*)**

Next Mode

RTR Echo configuration: **XSR(config-rtr-echo-*xx*)**

Example

The following example sets the RTR type and acquires RTR Echo mode:

```
XSR(config-rtr-57)#type echo protocol ipIcmpEcho 192.168.57.3
XSR(config-rtr-echo-57)
```

RTR-mode Commands

rtr

This command creates a Response Time Reporter (RTR) entry. The following are sub-commands:

- **rtr owner** registers the RTR administrator. Go to page 1-43 for the command description.
- **rtr schedule** configures when an RTR entry will be run. Go to page 1-44 for the command description.

Syntax

rtr *operation-id*

operation-id	Measurement ID number, ranging from 1 to 2,147,483,647.

Mode

Global configuration: **XSR(config)#**

Next Mode

RTR configuration: **XSR(config-rtr-*xx*)#**

Example

The following command configures RTR entry *1* and acquires RTR mode:

```
XSR(config)#rtr 1
XSR(config-rtr-1)#
```

rtr owner

This command registers the Response Time Reporter (RTR) administrator (owner).

Syntax

rtr owner {*owner-name*}[*ipAddress*][**quota** *quota*][**email** *email*][**sms** *sms*]

owner-name:	Owner's name which is case sensitive and must contain no spaces.
ipAddress	IP address of the management entity.
quota	Maximum number of records for this owner in the Enterasys service level reporting MIB history table, ranging from 1 to 10,500.
email	Owner's Email address.
sms	Owner's SMS phone number. It *must not* contain a space.

Mode

Global configuration: **XSR(config)#**

Default

Quota: 700

Example

The following example registers the RTR owner:

```
XSR(config)#rtr owner operator1 192.168.57.5 email larrycurtis@enterays.com quota 1000
```

rtr schedule

This command schedules an Response Time Reporter (RTR) entry.

Syntax

rtr schedule *operation-id* [[**life** {*forever* | *lifetime*}] **start-time** {*hh:mm*:[*ss*][*month day* | *day month*] | *pending* | *now* | **after** *hh:mm:ss*}]

operation-id	Measurement ID number, ranging from 1 to 2,147,483,647.
lifetime	Entry lifespan, ranging from 1 to 2,147,483,647 seconds.
hh:mm:ss	Time in hours, minutes and seconds.
day	Day of the month.
month	Month of the year.
pending	Operation will not begin. This state is meaningful when used by SNMP. After an entry is scheduled, all supported metrics meaningful to the protocol type will be measured.

Mode

Global configuration: **XSR(config)#**

Default

pending

Example

The following example schedules the RTR measurement *immediately*:

```
XSR(config)#rtr schedule 1 now
```

RTR Show Commands

show rtr operation-state

This command displays the current operational state of the Response Time Reporter (RTR).

Syntax

show rtr operation-state [*operation-id*]

operation-id	Measurement ID, ranging from 1 to 2,147,483,647.

Mode

EXEC configuration: **XSR>**

Sample Output

The following is sample output from the command:

```
XSR>show rtr operation-state 57
RTR Entry Number: 1
Number of Operations Attempted: 84
Timeout Occurred: FALSE
Operational State of Entry: INACTIVE
```

show rtr configuration

This command displays your configuration of the Response Time Reporter (RTR).

Syntax

show rtr configuration [*operation-id*]

operation-id	Measurement ID number, ranging from 1 to 2,147,483,647.

Mode

EXEC configuration: **XSR>**

Sample Output

The following is sample output from the command:

```
XSR>show rtr configuration
RTR Entry Number: 1
Owner: monitor
Tag: all metrics
Map: network in Peoria
Type of Operation to Perform: echo
Operation Frequency (seconds): 60
Operation Timeout (milliseconds): 5000
```

```
Status of Entry (SNMP RowStatus): active
Protocol Type: ipIcmpEcho
Target Address: 192.168.57.3
Source Address: 192.168.57.43
Request Size (data portion): 12
Life (seconds): 5000
Next Scheduled Start Time: Start Time already passed
Number of History Buckets kept: 15
```

show rtr history

This command displays the measurement history of the Response Time Reporter (RTR).

Syntax

show rtr [*operation-id*]

operation-id	Measurement ID number, ranging from 1 to 2,147,483,647.

Mode

EXEC configuration: **XSR>**

Sample Output

The following is sample output from the command:

```
XSR>show rtr history 57
Owner:  operator-toronto
Target Address: 1.1.1.1

NET HISTORY TABLE
Bucket     Sequence  TimeStamp         Delay      Packet
Entry      Number                      (ms)       Loss
1          96        11:2:1 Sept 1     3          FALSE
2          97        11:2:1 Sept 2     3          FALSE
3          98        11:2:1 Sept 3     3          FALSE
4          99        11:2:1 Sept 4     3          FALSE

AGGR HISTORY TABLE
Bucket     Sequence  TimeStamp         Average       Average Pkt      Jitter
Entry      Number                      Delay (ms)    Loss %           (ms)
1          11        10:42:1 Sept 1    3             0                0
2          12        10:52:1 Sept 2    3             0                0
3          13        11:22:1 Sept 3    3             0                0
```

2

Configuring T1/E1 and T3/E3 Subsystems

Observing Syntax and Conventions

The CLI Syntax and conventions use the notation described in the following table.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[x]	[ ] Square brackets indicate an optional parameter (*italic*)
[x | y | z]	[|] Square brackets with vertical bar indicate a choice of values
{x | y | z}	{ | } Braces with vertical bar indicate a choice of a required value
[x {y | z}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<*xx*>)	*xx* signifies interface type and number, e.g.: **F1**, **S2/1.0**, **D1**, **M57**, **G3. F** indicates a FastEthernet, and **G** a GigabitEthernet interface
Next Mode entries display the CLI prompt after a command is entered	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

T1/E1 & T3/E3 Commands

The following commands define T1/E1 /T3/E3 subsystem functionality:

- "T1/E1 & T3/E3 Commands" on page 2-55.
- "T1/E1 and T3/E3 Clear and Show Commands" on page 2-74.
- "Drop and Insert Commands" on page 2-80.

Note: The configuration commands for T1/E1 ports that occupy T3/E3 lines are the same commands that exist for T1/E1 NIM cards.

cablelength

For T3 controllers only

This command specifies the distance of cabling from the XSR to the network equipment for a T3 NIM card only.

Note: Although you can specify cable length from 0 to 450 feet, the XSR recognizes only two ranges: *0 to 224* and *225 to 450*. For example, entering 35 feet selects the 0 to 224 range. If you later change the cable length to 40 feet, there is no change because 40 falls within the 0 to 224 range. But, if you change the cable length to 350, the 225 to 450 range is selected. The actual length you enter is stored in the configuration file.

Syntax

```
cablelength feet
```

feet	Distance to set the cable length, ranging from 0 to 450 feet.

Syntax of the “no” Form

The *no* form of this command sets the cablelength to the default value:

```
no cablelength
```

Mode

Controller configuration: **XSR(config-controller *xx*)#**

Default

224 feet

Example

The following example configures the T3 controller in slot 1, card 2 with line source clocking, M13 framing, in channelized mode, and a cablelength of 225 feet:

```
XSR(config)#controller t3 1/2/0
XSR(config-controller<T3-1/2/0>)#channelized
XSR(config-controller<T3-1/2/0>)#clock source line
XSR(config-controller<T3-1/2/0>)#framing m13
XSR(config-controller<T3-1/2/0>)#cablelength 225
```

cablelength long

For T1 controllers only

This command decreases the pulse from the transmitter for long haul applications on T1 controllers only. In long haul applications (length of the haul longer than 655ft, CSU interface) the transmit pulse masks are optionally generated according to ANSI T1.403 to reduce crosstalk on

the received signals. This feature is provided by placing a transmit attenuator in the data path. This attenuation is selectable from **0**, **-7.5**, **-15**, or **-22.5 dB**.

Note: Long haul line build-out (LBO) compensates for the loss in decibels based on the distance from the device to the first repeater in the circuit. A longer distance from the device to the repeater requires that the signal strength on the circuit be boosted to compensate for loss over that distance. The ideal signal strength should be between -15 dB and -22 dB, which is calculated by adding the Telecom/PTT company loss + cable length loss + line build out. The lengthening or building out of a line is used to control far-end crosstalk. Line build-out attenuates the stronger signal from the customer installation transmitter so that the transmitting and receiving signals have similar amplitudes.

Syntax

cablelength long {*0db* | *-7.5db* | *-15db* | -22.5db}

0db	Number of decibels by which the transmit signal is lowered.
-7.5db	Number of decibels by which the transmit signal is lowered.
-15db	Number of decibels by which the transmit signal is lowered.
-22.5db	Number of decibels by which the transmit signal is lowered.

Syntax of the “no” Form

Use the *no* form of this command to return the LBO value to the default:

no cablelength long

Defaults

0 dB

Mode

Controller configuration: **XSR(config-controller<*xx*>)#**

Example

The following example sets the long haul LBO to –7.5 dB:

```
XSR(config)#controller t1 1/0
XSR(config-controller<T1-1/0>)#cablelength long -7.5db
```

cablelength short

For T1 controllers only

This command specifies the pulse shape of the transmit signals as defined in the ANSI T1.102 recommendation for short-haul applications.

These applications apply to haul lengths shorter or equal to 655‘(DSX-1 interface). This parameter is used to obtain an optimal pulse shape for external transformers. Five haul length ranges are defined, each with different pulse shaping settings: **0...133 ft** (0..40m), **133..266 ft** (40..81m), **266...399 ft** (81..122m), **399..533 ft** (122..162m), and **533..655 ft** (162..200m).

Syntax

cablelength short {*133* | *266* | *399* | *533* | *655*}

133	0 to 133 feet (cable length for short haul pulse shaping).
266	134 to 266 feet (cable length for short haul pulse shaping).
399	267 to 399 feet (cable length for short haul pulse shaping).
533	400 to 533 feet (cable length for short haul pulse shaping).
655	534 to 655 feet (cable length for short haul pulse shaping).

Syntax of the "no" form

The *no* form of this command returns the value to the default setting:

no cablelength short

Defaults

133 feet

Mode

Controller configuration: **XSR(config-controller<*xx*>)#**

Example

The following example sets the short haul LBO to 266 feet:

```
XSR(config)#controller t1 1/0
XSR(config-controller<T1-1/0>)#cablelength short 266
```

channel-group

For T1/E1 controllers only

This command specifies timeslots that map to channel-groups for T1/E1/ISDN-PRI data lines (for channelized/fractional T1/E1/ISDN-PRI services).

Timeslots and fractional/channelized T1/E1 groups allow multiple logical WAN interfaces to be created out of a single channelized T1 or E1 controller port. The logical interfaces created can have different encapsulation types – PPP, Frame Relay, etc. For each channel group (a fraction of a T1/E1/ISDN-PRI line), the following values must be set:

1. The channel group must be identified by a **channel group number**.
2. One or more **timeslots** of the T1/E1/ISDN-PRI line must be assigned to a particular channel group.
3. The base **speed** increment for the single channel can be specified in kilobits per second.

Syntax

channel-group *number* **timeslots** *range* [**speed** {**56** | **64**}]

number	Channel-group number, ranging from 0 to 23 for T1 and 0 to 30 for E1 data lines.

range	Assigns one or more timeslots or a range of timeslots to a channel group, ranging from 1 to 24 for T1 and 1 to 31 for E1.
speed	Line speed of the T1/E1 link in kilobits per second.

Syntax of the "no" Form

Use the *no* form of the command to remove a channel group:

no channel-group *number*

Defaults

Speed: 64 kbps for both T1 and E1 controllers.

Mode

Controller configuration: **XSR(config-controller<*xx*>)#**

Example

The following example issues the **channel-group** command for T1 controller configuration. Two channels are created – the first creates group number *0* with timeslots *1* to *10*; the second creates group number *1* with timeslots *11* to *20*, both with default speeds of *64* kbps.

```
XSR(config)#controller t1 1/0
XSR(config-controller<T1-1/0>)#description T1 for Acme
XSR(config-controller<T1-1/0>)#framing esf
XSR(config-controller<T1-1/0>)#linecode b8zs
XSR(config-controller<T1-1/0>)#channel-group 0 timeslot 1-10
XSR(config-controller<T1-1/0>)#channel-group 1 timeslot 11-20
```

clock source

This command defines the clock source for a T1/E1 or T3/E3 line. It is needed because of synchronous transmission of data on digital interfaces as in the case of T1/E1 or T3/E3 lines. The clock source sets the required timing synchronization between the transmitter and receiver using *line* and *internal* settings.

Syntax

clock source {*line* | *internal*}

line	Clock derived from the T1/E1 or T3/E3 line provider.
internal	Clock from a chip on the T1/E1 or T3/E3 controller card.

Syntax of the "no" Form

The *no* form of this command returns the value to the default setting:

no clock source

Default

Line

Mode

Controller configuration: **XSR(config-controller<*xx*>)#**

Examples

The following example configures the *T1* controller on NIM *1*, port *0* (first port), with ESF framing, *B8ZS* line encoding and *line* source clocking:

```
XSR(config-controller<T1-1/0>)#framing esf
XSR(config-controller<T1-1/0>)#linecode b8zs
XSR(config-controller<T1-1/0>)#clock source line
```

This example set the *E3* controller in with *line* source clocking and a national reserved bit of *0*:

```
XSR(config-controller<E3-1/2/0>)#clock source line
XSR(config-controller<E3-1/2/0>)#national bit 0
```

controller

This command configures a T1/E1 or T3/E3 controller. You can invoke **controller** when a T1/E1 or T3/E3 NIM card is present on the XSR. This command automatically provides a full-rate channel group on port 0, by default, and acquires Controller mode in which additional commands defining clock source, framing, line encoding, and others must be executed to configure the controller. For T1/E1 controllers only, if you prefer to configure a channel other than 0, you can manually create a channel group using all timeslots and proceed with port configuration.

If no additional commands are specified in this mode, a default non-channelized port is created with default values.

Syntax

controller {**t1** | **e1** | **t3** | **e3**}{*slot/card/port*}
controller {**t1** | **e1** | **t3** | **e3**}{*card/port*}

t1	A T1 controller.
e1	An E1 controller.
t3	A T3 (44.736 Mbps) controller.
e3	An E3 (34.368 Mbps) controller.
slot	Sets the number of the slot in a system with multiple card slots. The motherboard is slot zero (0). Slot number 0 can be omitted.
card	Sets the NIM card number in the card slot (1 or 2)
port	Sets the number of the port on the slot or the port number on a NIM card, starting with zero. Valid choices are: - *First port in first NIM card:* 0/1/0 or simply 1/0. - *Second port in second NIM card:* 0/2/0 or simply 2/0.

Syntax of the "no" Form

The *no* form of this command deletes the defined controller:

no controller {**t1** | **e1**| **t3** | **e3**}{*slot/card/port*}
no controller {**t1** | **e1**| **t3** | **e3**}{*card/port*}

Mode

Global configuration: **XSR(config)#**

Next Mode

Controller configuration: **XSR(config-controller<*xx*>)#**

Default

Full rate

Examples

The following example sets the *T1* NIM on board *1*, port *0* (first port) and maps timeslots to the channel group. Also, it assigns an IP interface, sets PPP encoding and enables Serial port *1/0*:

```
XSR(config-controller)#controller t1 1/0
XSR(config-controller<T1-1/0>)#clock source line
XSR(config-controller<T1-1/0>)#framing esf
XSR(config-controller<T1-1/0>)#channel-group 0 timeslots 1,3-5,8
XSR(config-controller<T1-1/0>)#no shutdown
XSR(config)#interface serial 1/0:0
XSR(config-if<S1/0:0>#ip address 10.1.11.2 255.255.255.0
XSR(config-if<S1/0:0>#encapsulation ppp
XSR(config-if<S1/0:0>#no shutdown
```

This example sets the *E1* NIM on board *1*, port *0* (first port) to use all channels at full rate:

```
XSR(config-controller)#controller e1 1/0
XSR(config-controller<E1-1/0:0>)#no shutdown
XSR(config)#interface serial 1/0:0
XSR(config-if<S1/0:0>#ip address 10.11.44.3 255.255.255.0
XSR(config-if<S1/0:0>#encapsulation ppp
XSR(config-if<S1/0:0>#no shutdown
```

The following example configures the *T3* controller in slot *1*, card 1:

```
XSR(config)#controller<T3-1/1/0>)
XSR(config-controller<T3-1/1/0>)#clock source line
```

crc

For T1/E1 controllers only

This command sets the length of the Cyclic Redundancy Check (CRC) per channel group. CRC length can be set to 16 or 32 bits of the Frame Check Sequence (FCS). A 32-bit CRC provides more powerful error detection but adds overhead. Both receiver and sender must use the same setting.

Syntax

crc {*16* | *32*}

16 or 32	CRC size in bits per channel group or fractional link (port).

Syntax of the "'no" Form

The *no* form of this command returns to the default setting:

no crc

Default

16

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

This example enables the *32-bit* CRC on the *T1* interface:

```
XSR(config)#interface serial 1/0:2
XSR(config-if<S1/0:2)#crc 32
```

description

This command identifies the T1/E1 or T3/E3 controller. The description string provides a more descriptive name/comment for a particular T1/E1 or T3/E3 line. This parameter can be a string value of arbitrary length (max 80 characters). In all statistics reporting, this value identifies the T1/E1 or T3/E3 line in a more descriptive way. This command is functional for all serial interfaces.

Syntax

description *"string"*

"string"	Comment (up to 80 characters) describing the T1/E1 or T3/E3 controller. Quotations are *mandatory.*

Syntax of the "no" Form

The no form of the command deletes the description:

no description

Mode

Controller configuration: **XSR(config-controller<*xx*>)#**

Examples

The following example configures the T1 controller in board (NIM care) *1*, port *0* (first port), with *ESF* framing, *B8ZS* line encoding and *line* source clocking with a description added:

```
XSR(config)#controller t1 1/0
XSR(config-controller<T1-1/0>)#framing esf
XSR(config-controller<T1-1/0>)#linecode b8zs
XSR(config-controller<T1-1/0>)#clock source line
XSR(config-controller<T1-1/0>)#description "Acme's T1"
```

The following example describes the T3 controller in slot *1*, card *2*:

```
XSR(config)#controller t3 1/2
XSR(config-controller<T3-1/2/0>)#description "T3 Up at ACME"
```

dsu mode

For T3/E3 un-channelized controllers only

This command configures an unchannelized sub-rate T3/E3 port to emulate a proprietary Data Service Unit (DSU) scheme. The XSR supports interoperability with a wide range of third-party DSU vendors.

Local DSU mode configuration *must* match the remote configuration, so you must know what type of DSU is connected to the remote port to determine if it interoperates with a T3 or E3 NIM. This command enables interoperability with providers using various T3 or E3 DSUs to provision the T3/E3 line.

Syntax

dsu mode {*digitallink* | *kentrox* | *larscom* | *adtran* | *verilink*}

digitallink	*Digitallink* mode connects the T3/E3 controller to a Digital Link, CISCO, or Quick Eagle DSU.
kentrox	*Kentrox* mode connects the T3/E3 controller to a Kentrox DSU.
larscom	*Larscom* mode links the T3 controller to a Larscom DSU.
adtran	*Adtran* mode connects the T3 controller to an Adtran T3SU 300.
verilink	*Verilink* mode connects the T3 controller to a Verilink HDM 2182.

Syntax of the "no" Form

The *no* form of this command sets the DSU mode to the default value:

```
no dsu mode
```

Mode

Controller configuration: **XSR(config-controller *xx*)#**

Example

The following example configures the T3 controller in slot 1, card 2 with line source clocking, M13 framing, in unchannelized mode, with a cable length of 250 feet, and DSU interoperability mode set to an Adtran DSU:

```
XSR(config)#controller<T3-1/2/0
XSR(config-controller<T3-1/2/0>)#no channelized
XSR(config-controller<T3-1/2/0>)#clock source line
```

```
XSR(config-controller<T3-1/2/0>)#framing m13
XSR(config-controller<T3-1/2/0>)#cablelength 250
XSR(config-controller<T3-1/2/0>)#dsu mode adtran
```

dsu bandwidth

For T3 controllers only

This command specifies the peak allowable bandwidth used by the T3/E3 port. DSU bandwidth configuration must match the remote configuration and it is important that you know the bandwidth value set on the remote port. For example, if you reduce the bandwidth to 7,000 kbps on the local port, you must do the same on the remote port. This command reduces bandwidth by padding the T3/E3 frame.

For E3 ports in bypass framing mode, DSU bandwidth defaults to 34,368 kbps.

Even though the XSR lets you configure a continuous range of bandwidths in sub-rate modes, vendors support bandwidths only in certain values. So, the XSR sets the user-configured bandwidth to the closest vendor-supported bandwidth (refer to Table 2-1) and a message displayed showing the new bandwidth. Use the **`show controller`** command to view the vendor-supported bandwidth the XSR sets.

Note: DSU bandwidth is configurable only for an *unchannelized* T3/E3 port.

Table 2-1 Vendor DSU Bandwidth

DSU Mode	DSU	Bandwidth Range (kbps)	Step Size (kbps)
digitallink	Digital Link, Quick Eagle, Cisco	300-44210 (T3), 358-34010 (E3)	300.746 (T3), 358 (E3)
kentrox	Kentrox	1500-35000/44210 (T3), 1000-24500/34010 (E3)	500 (T3/E3)
larscom	Larscom	3100-44210 (T3)	3158 (T3)
adtran	Adtran	75-44210 (T3)	75.186 (T3)
verilink	Verilink	1500-44210 (T3)	1579 (T3)
none	No DSU	44210 (T3) 34099 5	Fixed full rate

Syntax

`dsu bandwidth` *`bandwidth`*

`bandwidth`	Peak bandwidth allowed for the selected DSU, ranging from 1 to 44,210 kbps (T3) and 1 to 34,100 kbps (E3).

Syntax of the “no” Form

The *no* form of this command sets the DSU bandwidth to the default:

`no dsu bandwidth`

Mode

Controller configuration: **XSR(config-controller *xx*)#**

Default

- T3: 44,210 kbps (full-rate)
- E3: 34,099.5 kbps (full-rate)

Example

The following example configures the *T3* controller in slot *1*, card *2* with *line* source clocking, *M13* framing, in *unchannelized* mode, with a cable length of *250*, DSU interoperability mode set to a *Kentrox* DSU, and a DSU bandwidth of *44,210* kbps:

```
XSR(config)#controller t3 1/2/0
XSR(config-controller<T3-1/2/0>)#no channelized
XSR(config-controller<T3-1/2/0>)#clock source line
XSR(config-controller<T3-1/2/0>)#framing m13
XSR(config-controller<T3-1/2/0>)#cablelength 250
XSR(config-controller<T3-1/2/0>)#dsu mode 1
XSR(config-controller<T3-1/2/0>)#dsu bandwidth 44210
```

e-bit-reset

This command sets the E-bit in the E1 frame to zero while the port is in an asynchronous state.

Syntax

```
e-bit-reset
```

Syntax of the “no” Form

The *no* form of this command negates the E--bit reset:

```
no e-bit-reset
```

Mode

Controller configuration: **XSR(config-controller)#**

Example

The following example resets the E-bit on the E1 controller:

```
XSR(config-controller<E1-1/2/0>)#
```

equipment

For T3/E3 controllers only

This command configures the T3/E3 controller as network or customer equipment and operates according to the T1.403 ANSI standard, allowing equipment configured as network equipment to disregard network loopback commands from the far-end device.

Note: Since remote loopback requests are available only when C-bit framing is invoked for a T3 port, the **`equipment`** command is useful only when framing is set to C-bit.

Syntax

`equipment` {*`customer`* | *`network`*} **`loopback`**

`customer`	Controller set as *customer* equipment. It *allows* a remotely activated (feac) payload loop from the T3 line.
`network`	Controller set as *network* equipment. It *disallows* remotely activated (feac) payload loop from the T3 line.

Syntax of the “no” Form

The *no* form of this command sets the equipment value to its default:

```
no equipment
```

Mode

Controller configuration: **`XSR(config-controller)#`**

Default

Customer equipment

Example

The following example sets the T3 controller in slot *1*, card *2* as *network* equipment:

```
XSR(config-controller<T3-1/2/0>)#equipment network loopback
```

framing

This command sets the T1/E1 or T3/E3 framing type. Framing must match between the circuit provider and the T1/E1 or T3/E3 interface with the circuit provider determining which framing type is required.

Framing type defines the type and format of the transmission frame for T1 or E1 lines. T1 lines have two frame formats: **SF** (Super Frame, D4, F12) and **ESF** (Extended SF). E1 lines have these frame formats: **CRC4** (multiframe) and **NO-CRC4** (double frame).

For unchannelized T3 ports, the C-bit framing format is available with M13 as an option. For both channelized and unchannelized E3 ports, the G751 frame format is available. Also, the bypass framing format specifies that the G.751 framing format will be bypassed.

Note: The C-bit T3 parity framing format is an enhancement of the original M13 format. The main difference is the C-bit framing format always stuffs the first bit of the 8th block in each sub-frame. So, in C-bit format, C-bits permit greater management and performance functions on the M frame.

Syntax

framing {**sf** | **esf**} (*T1*)
framing {**crc4** | **no-crc4**} (*E1*)
framing {**c-bit** | **m13**} (*T3*)
framing {**g751** | **bypass**} (*E3*)

sf	T1 frame type set to Super Frame (D4, F12).
esf	T1 frame type set to Extended Super Frame.
c-bit	T3 frame type set to C-bit.
m13	T3 unchannelized frame type set to M13.
crc4	E1 frame type set to CRC4 frame.
no-crc4	E1 frame type set to no CRC4 frame.
g751	E3 frame type set to G.751.
bypass	E3 frame type set to be bypassed. Unchannelized implied.

Syntax of the "no" Form

Return to the default framing setting by using the *no* form:

```
no framing
```

Defaults

- T1: *ESF*
- E1: *CRC4*
- T3: *c-bit*
- E3: *g751*

Mode

Controller configuration: **XSR(config-controller<*xx*>)#**

Example

The following example configures the *T1* controller on NIM card *0*, port 0, with *ESF* framing:

```
XSR(config)#controller t1 1/0
XSR(config-controller<T1-1/0>)#framing esf
```

This example sets the *T3* controller with *line* source clocking, M23 framing, and *channelized* mode:

```
XSR(config-controller<T3-1/2/0>)#channelized
XSR(config-controller<T3-1/2/0>)#clock source line
XSR(config-controller>T3-1/2/0>)#framing m13
```

interface serial

This command configures the Serial interface automatically created by the **controller** command in conjunction with T1/E1 and T3/E3 NIM operations. The T3 module offers channels to PPP and Frame Relay protocol stacks. T3/E3 Serial channels are configured and monitored similar to serial channels provisioned via T1/E1 and serial NIMs. For full and sub-rate T3 or E3 mode, the port and channel setting is 0 only.

Syntax

interface serial {*slot* | *card* | *port0* | *channel0*}

slot	Slot number of a system from 0 to 6 card slots. The motherboard is slot zero. If the slot number is 0, it can be omitted.
card	Defines NIM card number in the card slot: 1 or 2.
port	Defines the port number on the slot or the port number on a NIM card, from 0 to 3.

Mode

Interface configuration: **XSR(config-if<S*xx*>)#**

Example

The following example configures Serial interface *2/0*:

```
XSR(config)#interface serial 2/0
XSR(config-if<S2/0>)#
```

international bit

For E3 controllers only

This command sets bits 6 and 8, respectively, of set II in the E3 frame.

Syntax

international bit {*0* | *1*}{*0* | *1*}

0 | *1*	Value of the *first* international bit in the G.751 frame.
1 | *1*	Value of the *second* international bit in the G.751 frame.

Syntax of the “no” Form

The *no* form of this command sets the international bits to the default:

```
no international bit
```

Mode

Controller configuration: **XSR(config-controller *xx*)#**

Default

- First international bit: 0
- Second international bit: 0

Example

The following example configures the *E3* controller in slot *1*, card *2* with *line* source clocking and international bits of *0* and *0*:

```
XSR(config)#controller e3 1/2/0
XSR(config-controller<E3-1/2/0>)#clock source line
XSR(config-controller<E3-1/2/0>)#international bit 0 0
```

invert data

For T1/E1 controllers only

This command inverts the data stream. Data inversion is a method of avoiding excessive zeroes that is superseded by the use of B8ZS line encoding. However, in cases where the network or remote node does not support this type of line coding, data belonging to an HDLC stream can be inverted to satisfy requirements of the line.

Syntax

```
invert data
```

Syntax of the "no" Form

Disable inverting the data stream by using the command's *no* form:

```
no invert data
```

Default

Data is *not* inverted.

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example enables data inversion on the full-rate *T1* interface in NIM card *1*, port *0*:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#invert data
```

linecode

For T1/E1 controllers only

This command defines the encoding type for T1/E1/ISDN-PRI lines. Configuration must match the required setting of the service provider. The service provider determines which line encoding type is required. The following three encoding types can be configured:

- *AMI* (Alternate Mark Inversion)
- *B8ZS* (Bipolar 8 Zero Substitution – T1 only)
- *HDB3* (High-density Bipolar 3 – E1 only)

Syntax

linecode {*ami* | *b8zs* | *hdb3*}

ami	Alternate Mark Inversion (AMI) line encoding.
b8zs	Bipolar 8 Zero Substitution (B8ZS) line encoding. Used for T1 controllers *only*.
hdb3	High-Density Bipolar 3 (HDB3) line encoding. Used for E1 controllers *only*.

Syntax of the "no" Form

Return to the default linecode setting by using the *no* form:

no linecode

Defaults

- T1 line: B8ZS
- E1 line: HDB3

Mode

Controller configuration: **XSR(config-controller<*xx*>)#**

Example

This example sets the *T1* controller with *ESF* framing, and *B8ZS* line encoding:

```
XSR(config)#controller t1 1/0
XSR(config-controller<T1-1/0>)#framing esf
XSR(config-controller<T1-1/0>)#linecode b8zs
```

loopback

For T1/E1 controllers only

This command implements loopback tests on a T1/E1/ISDN-PRI subsystem. Typically, it is used for diagnostic purposes although you can configure an IP address as a loopback interface as shown in the example. If you configure a loopback address for the XSR, it will be used as the Router ID. If there is no loopback address defined, the Router ID is the highest non-zero IP address of existing configured and active interfaces.

When a T1/E1/ISDN-PRI line malfunctions, one troubleshooting option is to perform various loopback tests, for instance, isolating pieces of the link to test separately. Loopback testing should begin on the local router and proceed to testing the service/network provider. Be aware that all loopback testing is intrusive, and while loopback tests run, data transfers over the link are barred.

Syntax

loopback {*diagnostic* | *local* {*line* | *payload*}}

diagnostic	Loops the outgoing transmit signal back to the receive signal. Use the **show t1/e1 controller** command to check if loopback is set. Use **show interface serial** to verify that the channel groups are looped back.

local line	Local loopback mode loops the entire bandwidth of the T1/E1/ISDN-PRI line toward the network. Use external equipment to verify that the T1/E1/ISDN-PRI port is connected to the line.
local payload	Same as Local line, it merely loops back the T1 payload, that is, the XSR generates framing at 1.536 MBytes/sec.

Syntax of the "no" Form

no loopback

Default

Disabled

Mode

Controller configuration: **XSR(config-controller<*xx*>)#**

Examples

The following example initiates a *local* loopback test:

```
XSR(config)#controller t1 1/0
XSR(config-controller<T1-1/0>)#framing esf
XSR(config-controller<T1-1/0>)#linecode b8zs
XSR(config-controller<T1-1/0>)#channel-group 0 timeslot 1-24 speed 64
XSR(config-controller<T1-1/0>)#loopback local
```

The following example configures an IP address as a loopback interface:

```
XSR(config)#interface loopback 0
XSR(config-if<L0>)#ip address 193.23.24.1 255.255.255.255
XSR(config-if<L0>)#no shutdown
```

national bit

For E3 controllers only

This command sets the national bit in the E3 frame - bit 12.

Syntax

national bit {**0** | **1**}

0	Sets the national reserved bit to 0.
1	Sets the national reserved bit to 1.

Syntax of the "no" Form

The *no* form of this command sets the national bit to the default value:

no national bit

Mode

Controller configuration: **XSR(config-controller *xx*)#**

Default

1

Example

The following example configures the *E3* controller in slot *1*, card *2* with *line* source clocking and a national reserved bit of *0*:

```
XSR(config)#controller e3 1/2/0
XSR(config-controller<E3-1/2/0>)#clock source line
XSR(config-controller<E3-1/2/0>)#national bit 0
```

scramble

For T3/E3 controllers only

This command assists clock recovery on the receiving end of a T3/E3 port by randomizing the pattern of 1s and 0s carried in the physical layer frame. Randomizing the bits can prevent continuous, non-variable bit patterns, in other words, long strings of all 1s or 0s.

Several physical layer protocols rely on transitions between 1s and 0s to maintain clocking. Scrambling can prevent some bit patterns from being mistakenly interpreted as alarms. The following conditions must be met:

- Scrambling is used only for full-rate/sub-rate T3/E3 ports and they must be configured as *unchannelized* for scrambling to take affect.
- Remote and local T3/E3 scrambling configuration *must match*.
- For T3 controllers, all DSU modes support scrambling *except* Clear mode.
- For E3 controllers, only *Kentrox* mode supports scrambling.
- This value is configurable only on an *unchannelized* T3/E3 port.

Syntax

```
scramble
```

Syntax of the “no” Form

The *no* form of this command disables scrambling:

```
no scramble
```

Mode

Controller configuration: **XSR(config-controller *xx*)#**

Default

Disabled

Example

The following example configures the *T3* controller in slot *1*, card *2* with line source clocking, *M13* framing, in *unchannelized* mode, cablelength of *250*, DSU interoperability mode set to a *Kentrox* DSU, DSU bandwidth of *44210*, and scrambling enabled:

```
XSR(config)#controller t3 1/2/0
XSR(config-controller<T3-1/2/0>)#no channelized
XSR(config-controller<T3-1/2/0>)#clock source line
XSR(config-controller<T3-1/2/0>)#framing m13
XSR(config-controller<T3-1/2/0>)#cablelength 250
XSR(config-controller<T3-1/2/0>)#dsu mode kentrox
XSR(config-controller<T3-1/2/0>)#dsu bandwidth 44210
XSR(config-controller<T3-1/2/0>)#scramble
```

shutdown

This command disables disables a T1/E1/ISDN-PRI controller or the T3/E3 controller and all interfaces related to it. The command does not require any specific booting procedure and can be performed dynamically during system run-time. When the interface is created, it is disabled by default.

Disabling a T3/E/3 controller causes a T3 port to transmit:

- An Alarm Indication Signal (AIS) for M13 framing.
- An idle signal (for C-bit framing).

Ten seconds must elapse for alarms to clear after enabling a T3 port. Shutting down a controller causes an E3 port to transmit AIS.

Note: The AIS, also known as a *blue* alarm, is transmitted to notify the downstream device that an upstream line failure has occurred.

There is a short delay for alarms to clear after enabling an E3 port. It takes 10 seconds for alarms to clear after enabling a T3 port.

Syntax

```
shutdown
```

Syntax of the “no” Form

The *no* form of this command restores the previously configured T1/E1 controller and interface. Also, it re-enables a T1/E1/ISDN-PRI channel and associated serial interface:

```
no shutdown
```

Mode

Controller configuration: **`XSR(config-controller xx)#`**

Default

Disabled

Examples

The following example disables a *T1* controller:

```
XSR(config)#controller t1 1/0
XSR(config-controller<T1-1/0>)#shutdown
```

The following example re-enables a *T3* controller:

```
XSR(config)#controller t3 1/2/0
XSR(config-controller<T3-1/2/0>)#no shutdown
```

T1/E1 and T3/E3 Clear and Show Commands

clear controller

This command clears controller counters for individual T1/E1 or T3/E3 controllers. It clears only counters shown with **show** commands – all SNMP-related counters are not cleared. It does not reset or bring down the controller.

Syntax

clear controller {**t1** | **e1** | **t3** | **e3**}{*slot/card/port*}
clear controller {**t1** | **e1** | **t3** | **e3**}{*card/port*}

t1	T1 type controller.
e1	E1 type controller.
t3	T3 type controller.
e3	E3 type controller.
slot	Slot number of a system, ranging from 0 to 6. The motherboard is slot zero. If the slot number is 0, it can be omitted.
card	NIM card number in the card slot, ranging from 1 to 2.
port	Port number on a NIM card, ranging from 0 to 3.

Mode

Privileged EXEC: **XSR#**

Examples

The following example clears the T1 controller counters for board (NIM card) 1, port 0 (first port):

```
XSR#clear controller 1/0
Clear counters on controller 1/0 [confirm]
```

The following example clears the T3 controller in slot 1 and card 1:

```
XSR#clear controller t3 1/1/0
Clear counters on controller 1/1 [confirm]
```

show controllers

This command displays the status and statistics for any controller. The T1/E1, T3/E3, and ATM subsystems track various status and statistical parameters, including the current controller configuration. The command also displays Maintenance Data Link (MDL) information (received strings) if MDL is configured and framing is set to C-bit on T3 NIMs.

Notes: The network can remotely test XSR's T1 ports by placing them in loopback. If this occurs, the controller will change state to DOWN for the duration of the test even if it remains synchronized.

Statistics displayed with the **show controllers** command are reset every 24 hours. That is, once the port or line is created with the **controller** command, the 24-hour timer starts.

Syntax

show controllers {*interface-type*} *slot* | *card* | *port*
show controllers {*interface-type*} *slot* | *port*

interface -type	XSR interface type: ATM, BRI, ISDN, T1, E1, T3, E3, Fast/GigabitEthernet, or Serial.
slot	Slot number of a system from 0 to 6 card slots. The motherboard is slot zero. If the slot number is 0, it can be omitted.
card	NIM card number in the card slot: 1 or 2.
port	Port number on the slot or the port number on a NIM card, from 0 to 3.

Mode

Privileged EXEC: **XSR#**

Default

T3/E3: Short display

Sample Output

This command displays T1 controller statistics with two channel-groups:

```
T1 0/2/1 is Admin Up and Oper Up.
T1 with CSU Interface.
Applique type is Channelized T1.
Central Office (Network) loopback is set as line.
No alarms detected.
Loopback is set as none.
Cablelength long is 0db and Cablelength short is 133ft.
Framing is esf, Line Encoding is b8zs, Clock Source is line.
Description: None

Alarms Detected: None

Rx signal level -0.0DB (Accuracy:+/-3DB) [NULL string]

Bypass time slots table ( * data time slots on s/c/0 and s/c/1):
                          1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
```

```
          1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
Rx ABCD * * * * * * F F 0 * F F F F F F F F F F F F F F

Channel 1:
        Timeslots 1,2,3,4,5,6,7,8,9,10
        64kbps Base rate
Channel 2:
        Timeslots 12,13
        56kbps Base rate

Data in current interval (502 seconds elapsed):
   0 Line Code Violations
   0 Path Code Violations
   0 Slip Seconds
   0 Frame Loss Seconds
   0 Line Error Seconds [string]
   0 Degraded Minutes
   0 Errored Seconds
   0 Bursty Error Seconds
   0 Severely Error Seconds
   0 Unavailable Seconds

Total Data (last 24 hours):
   0 Line Code Violations
   0 Path Code Violations
   0 Slip Seconds
   0 Frame Loss Seconds
   0 Line Error Seconds
   0 Degraded Minutes
   0 Errored Seconds
   0 Bursty Error Seconds
   0 Severely Error Seconds
   0 Unavailable Seconds
```

The following line is added to the output if loopback is set as line:

```
Central Office (Network) loopback is set as line.
```

The following is a partial example of the output from a T3 NIM:

```
XSR#show controllers t3 0/1/0
T3 0/1/0 is Admin Down and Oper Down.
Appliqué type is Un channelized T3.
Loopback is set as none.
Equipment is set as customer.
MDL transmission is disabled.
Cablelength range is 0-224 feet.
Framing is C-BIT, Clock Source is Line.
Scramble is disabled.
DSU is set to None with bandwidth 44210 kbps.
Description: None

FEAC codes Received:
```

```
 Latest          II               III              IV
No Code        No Code          No Code          No Code

Alarms Detected:
  LOS    LOF    TxAIS   RxAIS   TxRAI   RxRAI   LOOP   PayLd
   X      X       X

24 Hour Statistics cleared: MAY 04 22:33:47
Current time: MAY 04 22:34:13

Interval LVC  PCV  CCV  PES  PSES SEFS UAS  LES  CES  CSES
Total    4352 0    0    2    2    2    2    2    2    2
Current  4352 0    0    2    2    2    2    2    2    2
 ( 28s)

Note:
The 24 hour statistics is applied differently based on the selected farming type,
the following table marks the valid fields by a *

         LCV PCV CCV PES PSES SEFS UAS LES CES CSES

T3 C_bit  *   *   *   *    *   *    *   *    *   *
T3 M13    *   *   -   *    *   *    *   *    -   -

E3 G751   *               SES  *    *   *
E3 Bypass *                         *
```

Parameter Descriptions

Rx signal level -0.0DB (Accuracy:+/-3DB) [string]	String values can be: • NULL string: port locked on the signal; range 0 to 43.4 • "not valid": port could not lock on the signal 0 to 43.4 • "high noise floor": port locked on the signal, but signal is noisy 0 to 43.4. This line determines if the port is connected to a valid T1/E1 signal. The port will not function if the signal is "not valid" and will act unpredictable if it is "high noise floor". The line displays only if the Drop&Inset NIM is configured for data and voice mode. If it is used in data mode, it will not display.
1 1 1 1 1 1 1 1 1 1 2 2 2 2 2	Time slot number TENs.
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4	Time slot number units.

*Rx ABCD * * * * * * F F 0 * F F F F F F F F F F F F F F*	Time slot that bypasses between port 0 and 1 carry Channel Associated Signaling (CAS). CAS signaling comprises four bits: Bit A, C, C and D. This line shows CAS signaling for each voice channel by which you can determine channel status based on the current CAS value. It is a debug aid. Channels marked with an asterisk (*) are read as follows: • 1 - On the displayed port, timeslot 10 is used for data and is marked with an asterisk (*) • 2 - On the complementary port, (the other port of the card) timeslots 1 through 6 are used for data. • 3 - All time slots not used for data on neither port are bypassed between the two ports and their CAS -displayed.
T3 0/1/0 is up	T3 controller in slot 0 is operating. The controller's state can be *up*, *down*, or *administratively down*. Loopback conditions are shown as (Locally looped) or (Remotely Looped).
Applique type	Channelized or Non Channelized.
Alarms detected	Any alarms detected by the controller are displayed here. Any active alarm will bring the controller to *Oper Down* state. The YELLOW LED beside the port connector is ON for all physical alarms, but stays OFF for loopback modes. The following alarms are listed: • Transmitter is sending remote alarm (TxRAI). • Transmitter is sending TxAIS. • Receiver has loss of signal (LOS). • Receiver is getting RxAIS. • Receiver has loss of frame(LOF). • Receiver has remote alarm (RxRAI). • Receiver has no alarms (NONE). • Controller is set into a Payload Loop back (PayLd) from the network. • Controller is set locally or from the network into any type of Loopback (LOOP) from the network.
Network Line Loopback	• *None* - normal operation • *DS3 Line Loopback* (applicable for C-bit parity only)
MDL transmission is disabled	Status of the maintenance data link (either *enabled* or *disabled*).

FEAC code received	Displays the last 4 FEAC codes or commands that were received. Applicable for C-bit parity framing only, per ANSI T1.105-1995. This field are intended for T3 line debugging by carrier personal. Values (the last four codes are just displayed, subsequent codes will overwrite current ones) listed are as follows: • DS3 Eqpt. Failure (SA) • DS3 LOS • DS3 Out-of-Frame • DS3 AIS Received • DS3 IDLE Received • DS3 Eqpt. Failure (NSA) • Common Eqpt. Failure (NSA) • Multiple DS1 LOS • DS1 Eqpt. Failure (SA) • Single DS1 LOS • DS1 Eqpt. Failure (NSA) • No code is being received Command values are as follows: • Loopback Activate • Loopback Deactivate • DS3 Line • DS1 Line 1 to 28 (displayed but not acted upon) • DS1 Line All (displayed but not acted upon)
Framing is	Framing type on the controller: • C-BIT Parity • M13 • G.751 • Bypass
Line Code is	Line coding format on the controller: *B3ZS*
Clock Source is	Clock source on the controller: *Internal* or *Line*.
Line Code Violations (Valid for C-bit, M13, g751 & bypass)	A count of both Bipolar Violations (BPVs) and Excessive Zeros (EXZs) occurring over the accumulation period. An EXZ increments the LCV by one regardless of the zero string's length.
P-bit Coding Violation (Valid for C-bit & M13)	For all DS3 applications, a PCV error event is a P-bit parity error event. A P-bit parity error event is the occurrence of a received P-bit code on the DS3 M-frame that is not identical to the corresponding locally calculated code.
C-bit Coding Violation (Valid for C-bit)	For C-bit parity applications, the CCV is the sum of coding violations reported via the C-bits. For C-bit parity, it is the sum of CP-bit parity errors occurring during the accumulation interval.
P-bit Err Secs (Valid for C-bit & M13)	PES is a second with one or more PCVs, one or more Out-of-Frame defects, or a detected incoming AIS. This gauge is not incremented when unavailable seconds are counted.

P-bit Severely Err Secs (Valid for C-bit & M13)	PSES is a second with 44 or more PCVs, one or more Out-of-Frame defects, or a detected incoming AIS. This gauge is not incremented when unavailable seconds are counted.
Severely Err Secs (Valid for g751)	SES is a second in which more then 43 LCV were counted or one or more Out-of-Frame defects, or a detected incoming AIS. This gauge is not incremented when unavailable seconds are counted.
Severely Err Framing Secs (Valid for C-bit, M13 & g751)	SEFS is a second with one or more Out-of-Frame defects or a detected incoming AIS.
Unavailable Secs (Valid for C-bit, M13 & g751)	UAS are calculated by counting the period the interface is unavailable.
Line Err Secs	LES is a second with one or more code violations or one or more LOS defects.
C-bit Errored Secs (Valid for C-bit)	CES is a second with one or more C-bit code violations (CCV), one or more Out-of-Frame defects, or a detected incoming AIS. This gauge is not incremented when UASs are counted.
C-bit Severely Errored Secs (Valid for C-bit)	CSES is a second with 44 or more CCVs, one or more Out-of-Frame defects, or a detected incoming AIS. This gauge is not incremented when UASs are counted.

Drop and Insert Commands

These commands effect the operation of the T1/E1 Drop and Insert NIM.

drop-and-insert-group

This command, which takes no parameters, instructs the T1 controller to offer all its idle time slots not configured as part of a *channel-group* to the Drop and Insert (D&I) agent. The T1 controller thus operates in mixed Data/Voice mode.

For T1 lines, *robbed bit signaling* is used for Channel-Associated Signaling (CAS). Robbed Bit Signaling uses one bit of each timeslot for signaling every sixth frame. The XSR is configured in such a way that RBS is disabled for data timeslots (timeslots belonging to a channel group) and data can be passed at 64 or 56 Kbs.

When the command is issued for *both* T1 controllers on the NIM, time slots which are idle on both ports will be connected.

It is mandatory that the T1 port connected to the Central Office derive its timing from the *up stream* line and the port connected to the PBX supply timing to the *downstream* line.

Syntax

```
drop-and-insert-group [cas | clear]
```

cas	For use if the device downstream is a PBX using rob bit signalling. Entering no parameter is equivalent to entering the *no* command.
clear	For use if the device downstream handles data such as a Voice over IP.

Syntax of the “no” Form

The *no* form of this command removes Drop and Insert functionality:

```
no drop-and-insert-group
```

Mode

Controller configuration: **XSR(config-controller<*xx*>)#**

Default

cas

Example

This configuration instructs the XSR to terminate timeslots 1, 2, 3, 4, 5, 6 and 7 of controller T1 0/1/0 into a PPP channel and bypass the rest of the timeslots from T1 controller 0/1/0 to controller T1 0/1/1. controller port T0/1/0 is connected to the Central Office and controller port T0/1/1 is connected the the PBX down stream. Note that setting the clock source to *internal* is mandatory.

```
XSR(config)controller T1 0/1/0
XSR(config-controller<T1-0/1/0>)#drop-and-insert-group
XSR(config-controller<T1-0/1/0>)#channel group 0 timeslots 1,2,3-7 speed 56

XSR(config-controller<T1-0/1/0>)#clock source line
XSR(config-controller<T1-0/1/0>)#no shutdown
XSR(config-if<S0/1/0>)#interface serial 0/1/0
XSR(config-if<S0/1/0>)#encapsulation ppp
XSR(config-if<S0/1/0>)#no shutdown

XSR(config)#controller 0/1/1
XSR(config-controller<T1-0/1/0>)#drop-and-insert-group
XSR(config-controller<T1-0/1/0>)#no channel group 0
XSR(config-controller<T1-0/1/0>)#clock source internal
XSR(config-controller<T1-0/1/0>)#no shutdown
```

show controller

For Drop & Insert NIM only

This command, useful for debugging, lists the bypassed time slots between the two T1 controllers on the NIM and associated CASABCD signaling bits received. The Rx ABCD row displays the hex value of the CAS signaling bits received by the controller. Timeslots terminated in the XSR are marked with an asterisk (*). Those timeslots are used for data on ports 1 and/or 0. The bypass timeslot table will display only if the configuration is correct, that is, D&I is enabled on both ports and one of the ports employs internal clocking.This command may help debugging CAS voice calls.

Syntax

show controller t1 {*slot | card | 0/1*}

Example

This example shows port *0* using timeslot *10* for data and port *1* using timeslots *1 - 6* for data:

```
T1 0/1/0 is Admin Up and Oper Up.
T1 with CSU Interface.
```

```
Applique type is Fractional T1.
Loopback is set as none.
Cablelength long and short 0.
Framing is esf, Line Encoding is b8zs, Clock Source is line.
Description: None

Alarms Detected: None

Rx 0signal level -0.0DB (Accuracy:+/-3DB)
Bypass time slots table ( * data time slots on s/c/0 and s/c/1):
                          1 1 1 1 1 1 1 1 1 2 2 2 2 2
        1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
Rx ABCD * * * * * * F F 0 * F F F F F F F F F F F F F F

Channel 0:
        Timeslots 10
        64kbps Base rate

Data in current interval (300 seconds elapsed):
        0 Line Code Violations
        0 Path Code Violations
        8 Slip Seconds
        0 Frame Loss Seconds
        0 Line Error Seconds
        0 Degraded Minutes
        0 Errored Seconds
        0 Bursty Error Seconds
        0 Severely Error Seconds
        9 Unavailable Seconds
Total Data (Last 0 hours and 0 minutes):
        0 Line Code Violations
        0 Path Code Violations
        0 Slip Seconds
        0 Frame Loss Seconds
        0 Line Error Seconds
        0 Degraded Minutes
        0 Errored Seconds
        0 Bursty Error Seconds
        0 Severely Error Seconds
        0 Unavailable Seconds
```

3

Configuring the XSR Platform

Observing Syntax and Conventions

The CLI command syntax and conventions use the notation described in the following table.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[x]	[ ] Square brackets indicate an optional parameter (*italic*)
[x | y | z]	[|] Square brackets with vertical bar indicate a choice of values
{x | y | z}	{ | } Braces with vertical bar indicate a choice of a required value
[x {y | z}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<xx>)	*xx* signifies the interface, class map, policy map or other value you specify; e.g., **F1**, **G3**, **S2/1.0**, **<Your Name>**. **F** indicates a FastEthernet, and **G** a GigabitEthernet interface.
Next Mode entries display the CLI prompt after a command is entered.	
Sub-commands are displayed in ***red*** text.	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

Platform Commands

The following sets of commands define the platform subsystem software of the XSR:

- "Clock Commands" on page 3-84.
- "Crypto Key Commands" on page 3-85.
- "Other Platform Commands" on page 3-86
- "Platform Clear and Show Commands" on page 3-94.
- "File System Commands" on page 3-107.
- "Bootrom Monitor Mode Commands" on page 3-121.

Clock Commands

clock set

This command sets the current time of the Real Time Clock chip (software module clock). After resetting the XSR, you must manually set the clock.

Syntax

clock set hh:mm:ss wday mday month year

hh:mm:ss	Current time.
wday	Day of the week, ranging from 1 to 7. Sunday is 1.
mday	Day of the month, ranging from 1 to 31.
month	Month of the year. January is 1.
year	Year, ranging from 2000 to 2100.

Mode

Privileged EXEC: **XSR#**

Example

Set the clock to 2:59:59 p.m., Friday, October 7, 2002. Type the following:

```
XSR#clock set 14:59:59 06 07 10 2002
```

clock timezone

This command sets the time zone to reflect the local time and can be offset by up to 12 hours behind or 13 hours ahead of the Universal Time Clock (UTC) time as set for Greenwich Mean Time (GMT).

Syntax

clock timezone hh mm

hh	Number of hours offset (-12 behind to +13 ahead of GMT).
mm	Number of minutes offset (0 to 59).

Mode

Privileged EXEC: **XSR#**

Example

This example sets the time-zone 5 hours and 30 minutes behind UTC time (Eastern standard time):

```
XSR#clock timezone -5 30
```

Crypto Key Commands

crypto key master generate

This command generates a random master encryption key. When the command is entered, you are prompted to identify the previous master key. If you successfully identify it, the current secure data files are converted to use the new key. If not, you have the following options:

- Retry entering the previous key,
- Abort the key change,
- Remove the previous file set and enter a new key.

Note: This CLI command is not reflected in the *running-config*.

Syntax

```
crypto key master generate
```

Mode

Global configuration: **`XSR(config)#`**

Example

```
XSR(config)#crypto key master generate
```

crypto key master remove

This command removes the master encryption key. When entered, the command prompts you to identify the previous master key. If you successfully identify it, the current secure data files are removed. If not, you have the following options:

- Retry entering the previous old key,
- Abort the key removal process.

Syntax

```
crypto key master remove
```

Mode

Global configuration: **`XSR(config)#`**

Example

```
XSR(config)#crypto key master remove
```

crypto key master specify

This command allows you to specify a master encryption key. When entered, the command first prompts you to identify the previous master key. If you cannot identify it, you have the following options:

- Retry entering the previous key,
- Abort the key change,
- Remove the previous file set and enter a new key.

If you successfully identify a new key or proceed regardless of a correct response, you are prompted to specify a new key numbering 24 bytes. This new key will be rejected if it is identified as a *weak*, *semi-weak*, or *possibly weak* key. If you specify a valid new key, the current secure data files are converted to the new key.

Note: This CLI command is not reflected in the *running-config*.

Syntax

```
crypto key master specify
```

Mode

Global configuration: **`XSR(config)#`**

Example

```
XSR(config)#crypto key master specify
```

Other Platform Commands

cpu-utilization

This command enables the XSR to calculate the interval it spends on particular tasks and provides the utilization percentage per that task. CPU statistics are displayed using the **`show cpu-utilization`** command.

Syntax

```
cpu-utilization
```

Syntax of the “no” Form

The *no* form of this command disables CPU utilization reporting:

```
no cpu-utilization
```

Mode

Global configuration: **`XSR(config)#`**

Example

```
XSR(config)#cpu-utilization
```

debug processor

This command defines a method to force forwarding engine jobs to a specific CPU or allows the jobs to float between available CPUs.

Syntax

```
debug processor {number | job type | interface | mobility}
```

number	CPU: **0** or **1**.
job type	**Input**, **Output**, or **Protocol**.
interface	The specified interface.
mobility	**Fixed** (assign to a CPU and port) or **floating** (XSR assigns CPU and port).

Mode

Privileged EXEC: **XSR#**

Examples

The following example forces *CPU 0* to accept forwarding jobs input to *F1*:

```
XSR#debug processor 0 Input FE1 FIXED
Input Job for Interface FastEthernet 1 is now fixed to Processor #0
```

This example forces *CPU 1* to accept protocol forwarding jobs on interface *F2*:

```
XSR#debug processor 1 Protocol FE2 FIXED
Protocol Job for Interface FastEthernet 2 is now fixed to Processor #1
```

hostname

This command sets the system network name on the CLI prompt.

Syntax

```
hostname name
```

name	Name of the XSR that appears at the CLI prompt.

Syntax of the “no” Form

The *no* form of this command deletes the configured hostname:

```
no hostname
```

Mode

Privileged EXEC: **XSR#**

Default

The name that is stored in Bootrom.

Example

```
XSR#hostname XSR-1800
XSR-1800#
```

logging

This command enables/disables message logging at varying severity levels for specified destinations. Refer to *Appendix A* in the *XSR User's Guide* for a list of most router alarms and events. Normally, only HIGH severity alarms are logged to red flag critical events and those requiring operator intervention. The DEBUG alarm level is meant for maintenance personnel only.

The XSR may discard LOW and DEBUG level alarms if the system is too occupied to deliver them. The number of discarded messages is displayed by the following line in **`show logging`** command output:

```
Discards: high=0 medium=0 low=4 debug=22
```

The XSR supports as many as three Syslog servers, with logging severity levels separately configurable for each server. You can disable logging to individual Syslogs with the **`no logging`** ***`xxx.xxx.xxx.xxx`*** command.

LogGen Functionality

The *file* option permits logging to a persistent alarm file on a CompactFlash card for HIGH or MEDIUM alarms only. If no CompactFlash card is installed, persistent logging is not performed. The router copies messages from the logging buffer in RAM to the *cflash:* file *loggen* once per second. If power to the XSR is lost, the alarm history is preserved in *loggen*. When the XSR comes up again it copies the history from *loggen* back into the RAM buffer. The *entire* logging history is available including alarms before and after power-down.

The XSR's LogGen functionality declares a message flood if too many outstanding messages are reported by other software modules in the router. LogGen then temporarily quits reporting on the Console so users can keep access to the CLI. Messages are logged to the RAM buffer only, and are gradually reported to all other enabled destinations. The message flood ends when LogGen reduces the number of outstanding messages below the defined threshold.

Syntax

```
logging [console | buffered | monitor | snmp | A.B.C.D | A.B.C.D | A.B.C.D | file
| timestamp][level | local | utc][high | medium | low | debug]
```

`console`	Displays system logs to the console terminal.
`buffered`	Saves system logs to the router's RAM.
`monitor`	Displays system logs to current CLI Telnet session.
`snmp`	Saves system logs to a remote SNMP trap.
`A.B.C.D`	Up to three Syslog server IP addresses: see table in User Guidelines.
`level`	Sets logging level to High, Medium, Low or Debug. Enter the level immediately after the logging keyword to set that level for all destinations. Enter the level after a destination to specify that level only.
`file`	Logs data to a file on a CompactFlash card.
`high`	Sets system log to High level.

medium	Sets system log to Medium level.
low	Sets system log to Low level.
debug	Sets system log to Debug level.
timestamp	Sets time and date.
local	Sets timestamp to local time.
utc	Sets timestamp to the Universal Time Clock.

Syntax of the "no" Form

Use the *no* form of this command to disable the earlier configured service:

```
no logging [console | buffered | monitor | snmp | A.B.C.D | file | timestamp]
```

Mode

Global Configuration: **XSR(config)#**

Defaults

- File: *off*
- A.B.C.D.: *0.0.0.0* (no messages sent until an IP address is set)
- Logging level: *High* for all destinations

User Guidelines

The table below displays standard syslog error message types and definitions.

Message Type	Definition
0: Emergency	System is unusable
1: Alert	Action must be taken immediately
2: Critical	Critical conditions
3: Error	Error conditions
4: Warning	Warning conditions
5: Notice	Normal but signification condition
6: Info	Informational
7: Debug	Debug-level messages
8: Security	Security related messages

The XSR recognizes messages at four levels, described in the table below:

Priority Code = Facility Code *8 + Severity		
Severity	User Level Message (**Facility** = 1)	Security/Auth Message (**Facility** = 10)
High, severity = 2 (Critical)	10	82
Med, severity = 3 (Error)	11	83
Low, severity = 4 (Warning)	12	84

Debug, severity = 7 (Debug)	15	87

Examples

This example sets logging at *High* for the *console* with a *local* timestamp:

```
XSR#logging console high timestamp local
```

The following example sets a Low logging level for all destinations with a *UTC* timestamp:

```
XSR#logging low timestamp utc
```

This example sets persistent logging of High severity messages to *CFlash:* with a *local* timestamp:

```
XSR#logging file high timestamp local
```

The following example sets the logging timestamp to local time. For information about a related command, refer to **clock timezone** on (page 3-84):

```
XSR#logging timestamp local
```

The following example sets the logging timestamp to universal time:

```
XSR#logging timestamp utc
```

Sample Output

The following is a sample LogGen message:

```
<186>Jan 27 09:13:05 10.8.40.2 LOGGEN: Message Flood: Display disabled,messages
logged to History Buffer.
```

The following is sample output for a message flood by the show log history command:

```
XSR#show log history

Log history buffer: logging severity=HIGH; messages logged= 2 <186>Jan
27 09:13:07 10.8.40.2 LOGGEN: Message Flood: Display disabled, messages logged to
History Buffer.
```

netload

This command selects the Remote Auto Install (RAI) option upon reboot. When no **startup-config** file exists in the XSR, the system begins remote auto install processing by default.

Syntax

```
netload [persistent]
```

persistent	RAI does not cease looking for a config file over the network. Omitting this option permits RAI processing for 5 minutes, after which the XSR ceases RAI, exits and reads an existing **startup-config**.

Syntax of the “no” Form

The *no* form of this command disables **netload**:

```
no netload [persistent]
```

Mode

Global configuration: **XSR(config)#**

Examples

The following example selects a *5-minute* auto install:

```
XSR(config)#netload
```

The following example selects a *persistent* auto install:

```
XSR(config)#netload persistent
```

SNTP Commands

sntp-client

This command enables the SNTP client and sets the Simple Network Time Protocol (SNTP) primary and alternate server IP addresses. Once the XSR is configured, it sends a time request to the SNTP server every poll interval to update local time.

Note: Setting the SNTP Server IP address to 0.0.0.0 disables the SNTP client.

Syntax

```
sntp-client server A.B.C.D [A.B.C.D]
```

A.B.C.D	IP address of the primary SNTP server.
[A.B.C.D	IP of the alternate SNTP server. Set only if the primary SNTP server IP is set.

Syntax of the "no" Form

The *no* form of this command disables the SNTP client:

```
no sntp-client
```

Mode

Global configuration: **XSR(config)#**

Defaults

- Primary and alternate server IP address: 0.0.0.0
- SNTP client is disabled

Example

The following example sets the primary SNTP server IP address:

```
XSR(config)#sntp-client server 192.168.27.88
```

sntp-client poll-interval

This command configures the interval the SNTP client waits, when synchronized, before sending another time request to an SNTP server. The *poll-interval* is applied continuously after the client is first synchronized. If both primary and alternate servers are configured, polls are sent only to the first server, once this was detected to be active and only if this server becomes inactive will the client start polling the alternate server. A client declares a server inactive if no response is received to ten consecutive requests.

When the time is not synchronized after boot up, a resynchronization interval is used to send time requests to the server at fixed intervals of 60 seconds. A maximum of 10 such requests are sent in case no answer was received before the SNTP client decides this server is down. If an alternate server address is configured, requests are sent out to it. The resync interval is used instead of the polling interval to ensure the time is learned fairly quickly if the poll interval was set to a higher value. After initial synchronization, client requests are sent using the *configured* poll interval.

Syntax

sntp-client poll-interval [*value*]

Parameters

value	Poll-interval, ranging from 16 to 16284 seconds.

Mode

Global configuration: **XSR(config)#**

Default

512 seconds

sntp-server enable

This command enables the SNTP server.

Syntax

sntp-server enable

Mode

Global configuration: **XSR(config)#**

Default

Disabled

no sntp-server

This command disables the SNTP server.

Syntax

```
no sntp-server
```

Mode

Global configuration: **XSR(config)#**

show sntp

This command displays the current status of the SNTP server.

Syntax

```
show sntp
```

Output

```
XSR>show sntp
SNTP server      Stratum   #Polls    Last Receive
30.10.1.22       10        1         00:36:39      Active    Unicast
1.1.1.1          0         0         Never…                  Unicast

Client Status: Enabled
Server Status: Enabled
Poll Interval: 512
Server requests:  125
Current Time: 00:36:42-UTC-Tuesday, 30-MAR-2004
```

Parameter Descriptions

SNTP server 30.10.1.22	The IP address of the designated SNTP server.
Stratum	Level of the network where the clock is located. The primary stratum is generally considered at stratum 1. The XSR default stratum is 10.
#Polls	Sum of client requests to the SNTP server.
Last Receive	Hour, minute and second of the last client reply from the SNTP server.
Active	Whether the SNTP is in active state.
Unicast	SNTP server point-to-point mode.
Client Status	State of the SNTP client - enabled or disabled.
Server Status	State of the SNTP server - enabled or disabled.
Poll Interval	Interval in seconds between client requests to the SNTP server.
Server requests	Sum of client requests to the server.

Clock is synchronized, stratum 10, reference is <RTC or last synchronized reference>

Nominal freq is xxxxx Hz, actual freq is xxxx Hz, precision is 2**16

Reference time is 12345678.12345678 (01:01:01.123 EDT Mon Jan 1 2004)

Clock offset is 1.1234 msec, root delay is 123.12 msec

Root dispersion is 12.12 msec, peer dispersion is 1.12 msec

Platform Clear and Show Commands

clear counter processor

This command clears processor performance information. CPU utilization is averaged over an 8-second interval.

Syntax

```
clear counter processor
```

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#clear counter processor
```

clear fault-report

This command deletes the fault report from RAM.

Syntax

```
clear fault-report
```

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#clear fault-report
```

Sample Output

```
No fault report to clear.
```
or

```
Fault report cleared
```

clear logging

This command deletes all messages from the logging buffer in RAM.

Syntax

```
clear logging
```

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#clear logging
```

show buffers

This command displays platform memory statistics and is helpful in discovering where memory leaks exist in various XSR modules. Memory is allocated in increments no smaller than 64 bytes.

Syntax

```
show buffers
```

Mode

Privileged EXEC configuration: **XSR#**

Sample Output

```
XSR#show buffers

Common Buffer Pool Usage:
Pre-Allocated: 1000 for FE
               1000 for FE Frag
                512 for Eth1
                512 for Eth2
               1536 for 4 port T1E1 card 2 in slot 0
        Total: 4560 1696 byte buffers = 7733760 bytes

Used:    Eth2:  128 of   512 in use. 0 allocations denied.
     T1E1-0/2:  512 of  1536 in use. 0 allocations denied.
      FE Frag:    0 of   877 in use. 0 allocations denied.
      Fwd Eng:    0 of   877 in use. 0 allocations denied.
         Eth1:  128 of   512 in use. 0 allocations denied.

Free: Buffers: 3792. Extra Mblks: 500. FrameElements: 5000

Jumbo buffers:    8192  16384  32768  65536
Available:        8/ 8   4/ 4   2/ 2   1/ 1
```

```
Memory Block Allocation:
 Memory Options enabled: None.
-------------------------------------------------------------------------
       Size    Number    Number  Avg.Size  Max.Size  Number of      Size
     Carved    Carved    In Use    In Use   Request   Requests  Upgrade
-------------------------------------------------------------------------
         64      7012      6516        26        64   20254275        0
        128      6673      6637       104       128     629751        0
        288      2425      2389       249       288      20319        0
        512        33        26       417       512       5866        0
       1024        38        29       703      1024      15652        0
       2080        43        41      1362      2056     148677        0
       4096        29        17      2919      4096        597        0
       9216        20        18      6950      9188         22        0
      17408        13        12     14069     16856         15        0
      40960        10        10     25767     38916         10        0
      69632         5         5     62716     65604          5        0
     135168         4         4    117320    131072        138        0
     291104         1         1    270336    270336          1        0
     480000         0         0         0         0          0        0
     700000         1         1    628488    628488          1        0
    1560000         0         0         0         0          0        0
-------------------------------------------------------------------------
TotalBytes:  4965504  4817920   3831992          (64MB)
Overhead:     521824
Uncarved:   37914272
Max Heap:    1399088
```

Parameter Descriptions

Size Carved	Allocated pool sizes supported by the memory manager.
Number Carved	Sum of blocks carved in each pool shown in Column 1.
Number in Use	Sum of blocks currently in use in this pool. Every time you enter the **show buffers** command, this column's data will be marked with a plus (+) or negative sign (-). The + indicates the number in use has increased since you last entered the command. The - indicates the number in use has decreased since you last entered the command.
Average Size in Use	Average size of the actual requested allocation bytes.
Max Size Request	Largest allocation requested in this pool.
Number of Requests	Sum of times a memory was allocated within this block size.
Size Upgraded	Sum of instances a memory that could have fitted in this block size was actually allocated from a larger block size. This mechanism functions if the XSR is out of uncarved memory *and* block memory of this size. For example, you request 30 bytes of memory. The memory manager learns that there is no more uncarved memory, examines the 64-byte pool, and finds no more blocks in that pool either. Then the memory manager considers the 128-byte pool and may find some free blocks there. You will receive a pointer to one of blocks in the 128-byte pool.

Overhead	Sum of overhead bytes used for memory tracking, etc.
Uncarved	Sum of bytes available to be carved into desired blocks.
Max Heap	Sum of bytes that can be allocated from the heap.

show buffers i/o

This command displays summary I/O (data buffers, frame elements) memory usage statistics. Allocations are based on the hardware present in the XSR.

Syntax

```
show buffers i/o
```

Mode

Privileged EXEC configuration: **XSR#**

Sample Output

```
Common Buffer Pool Usage:
----------------------------------------------------------------
Pre-Allocated: 2000 for FE
               1000 for FE Frag
               2048 for Eth1
               2048 for Eth2
               2048 for Eth3
               1536 for serial card
        Total:10680*1696 byte buffers
                       *1796 (including overhead) = 19181280 bytes

Used: FE Frag:    0 of  1500 in use. 0 allocations denied.
      Fwd Eng:    0 of  3200 in use. 0 allocations denied.
              Eth2:  128 of   512 in use. 0 allocations denied.
     T1E1-0/2:  256 of   768 in use. 0 allocations denied.
      FE Frag:    0 of   880 in use. 0 allocations denied.
      Fwd Eng:    0 of   440 in use. 0 allocations denied.
         Eth1:  128 of   512 in use. 0 allocations denied.

Free: Buffers: 10680. Extra Mblks: 500. FrameElements: 5000

Jumbo buffers:    8192  16384  32768  65536
Available:        8/ 8   4/ 4   2/ 2   1/ 1
```

Parameter Descriptions

Common Buffer Pool Usage	One buffer pool exists for data buffers. These buffer blocks are pre-allocated as shown below: • *2000 for FE*: 2000 x 1696-byte buffers were pre-allocated for use by the Forwarding Engine. • *1000 for FE Frag*: 1000 x 1696-byte buffers were pre-allocated for use by FE Fragmentation. • *2048 for Eth1*: 2048 x 1696-byte buffers were pre-allocated for use by the Ethernet Driver for Ethernet Port 1. • *2048 for Eth2*: 2048 x 1696-byte buffers were pre-allocated for use by the Ethernet Driver for Ethernet Port 2. • *2048 for Eth3*: 2048 x 1696-byte buffers were pre-allocated for use by the Ethernet Driver for Ethernet Port 3. • *1536 for serial card*: 1536 x 1696-byte buffers were pre-allocated for use by the Serial NIM card. • *Total:10680*1696 byte buffers*: Total number of 1696-byte buffers that were pre-allocated. There are 100 bytes of overhead per buffer, so the actual amount of memory used is 10680 x 1796-bytes.
Used: FE Frag	• *0 of 1500 in use*. 0 of the 1500 peak allowed blocks are currently in use. • *0 allocations denied*. 0 requests for allocation were denied.
Fwd Eng	• *0 of 3200 in use*. 0 of the 3200 peak allowed blocks are currently used. • *0 allocations denied*. 0 requests for allocation were denied.
Free	• *Buffers: 10680*. Number of data buffers free now (all are free). • *Extra Mblks: 500*. Number of MBLKs (used to link multiple buffers) now free. • *FrameElements: 5000*: Number of Frame Elements (used to link multiple frames together) free now.
Jumbo buffers: 8192 16384 32768 65536:	Size of each Jumbo buffer which is used for temporary storage of large packets before fragmentation.
Available: 8/ 8 4/ 4 2/ 2 1/ 1:	(Available/Maximum) jumbo buffers. 8/8 indicates 8 available out of a maximum of 8 buffers. This example has every size with all buffers available.

show buffers malloc

This command displays summary Malloc (tables, configuration structure) area memory statistics.

Syntax

```
show buffers malloc
```

Mode

Privileged EXEC configuration: **XSR#**

Sample Output

```
Memory Block Allocation:
Memory Options enabled: None.
-------------------------------------------------------------------
       Size   Number Number Avg.Size  Max.Size  Number of Size
     Carved   Carved In Use In Use    Request   Requests  Upgrade
-------------------------------------------------------------------
         64   8132   8081   22        64        5960439   0
        128   10210  10209  98        128       18507     0
        288   2273   2241   252       288       8152      0
        512   19     15     441       512       302       0
       1024   22     20     718       1024      142       0
       2080   31     30     1391      2052      48        0
       4096   17     9      3185      4096      357       0
       9216   13     11     7673      9188      15        0
      17408   11     10     13358     16984     11        0
      40960   14     13     24725     40048     14        0
      69632   7      7      60418     65604     7         0
     135168   3      2      118344    131072    556       0
     291104   3      3      220710    270336    3         0
     480000   1      1      354400    354400    1         0
     700000   1      1      628488    628488    1         0
    1560000   1      1      1033920   1033920   1         0
-------------------------------------------------------------------
TotalBytes:  8039296  7775776   5725016 (128MB)
Overhead:     664256
Uncarved:   82346656
Max Heap:    1312224
```

Parameter Descriptions

Refer to the **show buffers** command.

show clock

This command shows current Universal Time Clock (UTC) set by Greenwich Mean Time (GMT).

Syntax

```
show clock
```

Mode

Privileged EXEC: **XSR#**

Sample Output

```
XSR#show clock
10:41:20-UTC-Wednesday,20-AUG-2003
```

If the time-zone is set up, **show clock** displays both UTC and local time:

```
XSR#show clock
15:22:52-UTC-Thursday,28-FEB-2002
10:22:52-LOCAL-Thursday,28-FEB-2002
```

show cpu-utilization

This command tracks current use of various CPU processes as a percentage of total CPU usage for the last five second, one minute, and five-minute intervals, and the number of times each process was called in total since the XSR was powered on. Also, CPU utilization is shown: the first percentage indicates total CPU usage, the second indicates the percentage of CPU time spent at the interrupt level, and remaining percentages are total CPU usage for 1- and 5-minute periods.

The command is a good diagnostic tool to measure which process is consuming the most CPU time and how strenuously the CPU is working as a whole. The XSR is operating normally if the CPU can satisfy advertised throughput levels at maximum capacity.

Be aware that this command draws on processor capacity at the expense of operational needs.

Syntax

```
show cpu-utilization
```

Mode

EXEC or Privileged EXEC: **XSR>** or **XSR#**

Default

CPU usage tracking is on by default.

Sample Output

```
XSR#show processes cpu
Process          Runtime(m)       5Sec     1Min     5Min     Invoked
PP                 0.00           0.01%    0.00%    0.00%    16302
RIP                0.00           0.01%    0.01%    0.01%      334
OSPF               0.00           0.02%    0.01%    0.01%      465
Idle               5.40           99.17%   99.24%   99.26%       0
Other              0.04           0.80%    0.74%    0.72%    26700
CPU utilization for five seconds: 14.53%/0.80%; one minute: 9.88%; five minutes: 8.20%
```

Parameter Description

Process	XSR task measured including Packet Processor (XSR forwarding engine), RIP and OSPF Processors, Idle (calculated processor idle time), and Other (all other tasks).
Invoked	Number of times a process has been called since the XSR was active.
CPU utilization	Total percentage of CPU being used at each interval.
14.53%/0.80%; one minute: 9.88%; five minutes: 8.20%	The first percentage indicates the total and the second indicates the percentage of CPU time spent at the interrupt level, followed by one and five minute percentages.

show fault-report

This command displays the fault report captured when the XSR experiences a system problem. It contains information that pinpoints the cause of the software failure. This data is highly technical and is intended only for the use of service support engineers to diagnose the problem.

The fault report can be viewed in Bootrom monitor mode or on the CLI.

If the XSR experiences a processor exception, the software captures a fault report and restarts automatically. Only the first fault report is saved in case of multiple failures in a special RAM area and is preserved if the XSR is re-booted but is lost if the XSR is powered down.

Note: The XSR can store one fault report only.

The fault report contains the following data relevant to the failure:

- Cause of processor exception
- Time stamp
- Contents of processor registers
- Operating system status
- Status of tasks, current task (e.g., crashed task)
- Contents of stacks (task stacks, interrupt stack)
- Status of one special task (packet processor by default)
- Code around the crash program counter
- Task message queues
- Memory management statistics
- Task stack traces for all tasks

Watchdog Fault Report

A fault report is also captured in case a catastrophic watchdog interrupt occurs. If the software does not refresh the watchdog for several seconds a watchdog fault report is captured and the XSR is warm-booted. You can then examine the fault report to analyze the problem.

Syntax

`show fault-report [0 | 1]`

0 | 1	CPU 0 or 1 on XSR 3000 Series *only*. If neither are specified, both fault reports display.

Mode

Privileged EXEC: **`XSR#`**

Example

```
XSR#show fault-report
```

Sample Output

The following is sample output from an XSR-3020 router:

```
Fault Report captured in node RouterName on  Sept 22, 2001 at  3:30:59pm
Fault: Data TLB Miss
Processor up-time = 1234 hours 59 minutes 59 seconds

Processor = PowerPC 405 GP
Exception Vector Number = 0x1100
PC=00012345  SP(r1)=00044444  LR=12345678  CTR=12345678
r0 =12345678  r1 =00044444  r2 =12345678   r3 =12345678
r4 =12345678  r5 =00044444  r6 =12345678   r7 =12345678
r8 =12345678  r9 =00044444  r10=12345678  r11=12345678
r12=12345678 r13=00044444  r14=12345678 r15=12345678
r16=12345678 r17=00044444  r18=12345678  r19=12345678
r20=12345678  r21=00044444  r22=12345678  r23=12345678
r24=12345678  r25=00044444  r26=12345678  r27=12345678
r28=12345678  r29=00044444  r30=12345678  r31=12345678
sprg0=12345678 sprg1=12345678 sprg2=12345678 sprg3=12345678
sprg4=12345678 sprg5=12345678 sprg6=12345678 sprg7=12345678
xer=12345678 msr=12345678 dccr=12345678 dcwr=12345678
iccr=12345678 sgr=12345678 sler=12345678 suor=12345678
bear=12345678 besr=12345678
ccr0=12345678 evpr=12345678 esr=12345678 dear=12345678
srr0=12345678 srr1=12345678 srr2=12345678 srr3=12345678

Crashed Task TCB:
004b19170 12345678 12345678 12345678 12345678 12345678 12345678 12345678
004b19180 12345678 12345678 12345678 12345678 12345678 12345678 12345678
etc.

Crashed Task Stack:
004276ae  12345678 12345678 12345678 12345678 12345678 12345678 12345678
004276be  12345678 12345678 12345678 12345678 12345678 12345678 12345678
004276ce  12345678 12345678 12345678 12345678 12345678 12345678 12345678
004276de  12345678 12345678 12345678 12345678 12345678 12345678 12345678

VxWorks Tasks:
NAME        ENTRY       TID      PRI  STATUS     PC     SP        ERRNO  DELAY
tExcTask    _excTask    4b19170   0    PEND     4276be  4b1908c   d0003     0
tLogTask    _logTask    4b14758   0    PEND     4276be  4b14670   d0003     0
tWdbTask    0x417cc4    4b10c08   3    READY    4276be  4b10ae4   d0003     0

tExcTask Control Block
004b19170 12345678 12345678 12345678 12345678 12345678 12345678 12345678
004b19180 12345678 12345678 12345678 12345678 12345678 12345678 12345678
etc.

tExcTask stack:
004276ae  12345678 12345678 12345678 12345678 12345678 12345678 12345678
```

```
004276be  12345678 12345678 12345678 12345678 12345678 12345678 12345678
004276ce  12345678 12345678 12345678 12345678 12345678 12345678 12345678
004276de  12345678 12345678 12345678 12345678 12345678 12345678 12345678
etc. for all tasks

End of fault report.
```

When the XSR is automatically rebooted after a crash it performs a warm start. The following message is logged:

```
11 May 29 22:20:59 TORONTO: System warm boot from crash
```

show logging

This command displays the current message logging settings including all possible logging destinations and their enabled message-levels.

Syntax

```
show logging
```

Mode

EXEC or Privileged EXEC: **XSR>** or **XSR#**

Example

```
XSR#show logging
```

Sample Output

The following example displays logging information on the XSR including three Syslog servers:

```
XSR#show logging
   Destination          Severity      Message Count
   Syslog: 10.10.10.20 medium        43
   Syslog: 10.10.10.30 low           78
   Syslog: 10.10.10.40 high          3
   Console              high          1630
   Monitor              high          1630
   Buffered             high          1630
   SNMP                 high          0
   File                 disabled      0
   Discards:            high=0 medium=0 low=0 debug=0
   Logging              timestamp UTC
```

show logging file

This command displays messages logged in the persistent logging file **loggen** on an optional CompactFlash card. This file stores data in the CFlash: directory if power to the XSR is lost. When the XSR comes up again it copies the history from *loggen* back into the RAM buffer. If no CompactFlash card is installed, persistent logging is not performed.

Syntax

```
show logging
```

Mode

EXEC or Privileged EXEC: **XSR>** or **XSR#**

Example

```
XSR>show logging file
```

Sample Output

The following example displays the logging file information:

```
XSR#show logging file
History of logging to file cflash:loggen
File logging disabled
File cflash:loggen does not exist.
```

show logging history

This command displays the contents of the logging history buffer.

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#show logging history
```

Sample Output

The following command displays logging history and severity levels:

```
Log history buffer: logging severity=MEDIUM+HIGH; messages logged=  8
<186>Feb 4 09:12:28 192.168.27.38 CLI: User: admin logged in from console
<186>Feb 4 09:10:56 192.168.27.38 CLI: CLI config mode released by startup-config
<186>Feb 4 09:10:56 192.168.27.38 ETH: Interface FastEthernet1, changed state to up
<186>Feb 4 09:10:56 192.168.27.38 CLI: CLI config mode locked by startup-config
<186>Feb 4 09:10:53 192.168.27.38 PLATF: System warm boot from cli
<11>May 29 22:20:59 TORONTO : System restarted
<12>May 29 22:25:59 TORONTO : Serial 0 changed state from up to down
```

show sntp

This command displays SNTP (Simple Network Time Protocol) setup and traffic statistics.

Syntax

```
show sntp
```

Mode

Privileged EXEC: **XSR#**

Sample Output

```
XSR#show sntp
Server IP:192.168.27.88
Poll Interval: 512
Sntp Requested: 1
Last Synced: 17:00:34-UTC-Sunday,26-JAN-2003
Current Time: 10:53:01-UTC-Monday,27-JAN-2003
```

show version

This command displays current XSR hardware and firmware data.

Syntax

```
show version
```

Mode

Privileged EXEC: **XSR#**

Sample Output

The following is example is output from an XSR-1805:

```
XSR#show version
Enterasys Networks Operating Software
Copyright 2002 by Enterasys Networks Inc.

Hardware:
  Motherboard Information:
    XSR-1800 ID: 9002854-02 REV0A
    Serial Number: 0000019876543210
    Processor: IBM PowerPC 405GP Rev. D at 200MHz
    RAM installed: 32MB
    Flash installed: 8MB on processor board, 16Mb compact flash
    CompactFlash: SunDisk SDP 5/3 0.6 has 32047104 bytes
    Real Time Clock
  I/O on Motherboard:
    FastEthernet 1
    FastEthernet 2 Rev 0
    H/W Encryption Accelerator Rev 1
    T1E1 has 4 channelized ports in NIM slot 1. Rev 0
    ISDN BRI has 2 ST ports in NIM slot 2. Rev 1
    Empty internal NIM slot 3

Bootrom:
    Version 2.03 Built Jul 28 2003, 11:35:07
```

```
Software:
    Version 5.5.1.3, Built May 16 2003, 14:31:56
    CLI revision 1.5
    Software file is "xsr1800.fls" with VPN; with Firewall
    XSR-1800 uptime is 33 days, 10 hours, 44 minutes.
```

The following example displays output from an XSR-3150:

```
XSR#show version
Enterasys Networks Operating Software
Copyright 2003 by Enterasys Networks Inc.

Hardware:
   Motherboard Information:
    XSR-3150 ID: 9002914-04 REV0A CPLD Rev 3
    Serial Number: 3646031700233215
    Processor: Broadcom BCM1250 Rev 2 at 600MHz
    PowerSupply1, PowerSupply2
   Fans 1 2 3 4 5 6 7 8
    CPU Temperature Max: 80C Current: 38C
    Router Temperature Max: 60C Current: 24C
   RAM: 512MB without interleave
         Memory Bus at 120MHz, CASL at 2.0
   Bootrom Flash: 4MB
   Filesystem Flash: 8MB
   CompactFlash not present
    Real Time Clock
   I/O on Motherboard:
    GigabitEthernet 1 2 3
    Encryption Hardware: not present
    Slot 0 card 1: Empty
    Slot 0 card 2: Empty

Bootrom:
    Version 1.5, Built Aug 26 2003, 13:23:16

Software:
    Version 6.0.0.0, Built Sep  7 2003, 16:06:27
    CLI revision 1.5
    Software file is "xsr3000.fls" with VPN; with Firewall.
    XSR-3150 uptime is 0 years, 4 days, 2 hours, 4 minutes, 6 seconds.
```

show whoami

This command displays identification data for a current terminal session.

Syntax

```
show whoami
```

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#show whoami
```

Sample Output

```
XSR#show whoami
Comm Server "Enterasys", current line at 9600bps.
```

File System Commands

The XSR employs an MS-DOS-compatible file system in Flash memory. The following commands are available.

boot system

This command creates a **boot-config** file to store the firmware file name of the active software image. This file name points to the firmware file loaded during system initialization in the following sequence:

1. The **boot-config** file is looked up in either **flash:** or **cflash:**
   - If **boot-config** is not found there, the router proceeds to Step 2.
   - If the file named in **boot-config** is not found, the router goes to Step 3.
2. If the default file (**xsr1800.fls or xsr3000.fls**) is not found, the router goes to Step 3.
3. An FTP/TFTP server as defined in network parameters of Bootrom mode is queried. If the image is not found in this remote location, initialization is suspended in Bootrom mode.

The command initiates a script requiring confirmation of your intention.

Syntax

```
boot system <newName.FLS>
```

Mode

Global configuration: **XSR(config)#**

Default

- *XSR1800.FLS* - for Series 1800 routers
- *XSR3000.FLS* - for Series 3000 routers

Note: A new software image file name must use the **.fls** extension. Optionally, you can modify a file with the **rename** command.

Examples

The following XSR 1800 Series example creates a **boot-config** file pointing to the firmware file name **VPN_XSR1800.fls**:

```
XSR(config)#boot system VPN_XSR1900.fls
```

The following example renames the **VPN_XSR1900.fls** file to match the Bootrom default file name. After entering the command, you are prompted by the following script:

```
XSR(config)#rename VPN_XSR1800.fls xsr1800.fls
Rename flash:VPN_xsr1800.fls to flash:xsr1800.fls(y/n) ? y
renaming file flash:VPN_xsr1800.fls -> flash:xsr1800.fls
XSR#
```

The following example renames the firmware file as part of an FTP/TFTP transfer. After entering the command, you are prompted by this script:

```
XSR-1800#copy tftp://192.168.37.162/c:\firmware\VPN_xsr1800.fls flash:xsr1800.fls
Copy 'c:\firmware\VPN_xsr1800.fls' from server as 'xsr1800.fls' into Flash(y/n) ? y
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Download from server done
File size: 3242460 bytes
XSR-1800#
```

cd

This command changes the current directory to **flash:** or **cflash:** on the XSR file system.

Syntax

cd [flash: | cflash:]

flash:	Default directory in Flash memory.
cflash:	Default directory in CompactFlash memory.

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#cd cflash:
```

copy <file>

This command copies a file to a new file which may reside in a local directory, **flash:** or **cflash:**, or on a remote TFTP server. You can omit the destination filename if new and source file names are identical. The XSR's MS-DOS-compatible file system of On-Board Flash (**flash:**) or CompactFlash (**cflash:**) memory. **Copy** initiates a script prompting your confirmation.

Syntax

copy source destination

The possible options are:

XSR#copy {flash:| *cflash*:**}[filename]{flash:**| *cflash*:**}[filename]**

XSR#copy {*flash*:| *cflash*:}[*filename*] *tftp*: [[[// *location*]/*directory*]/*filename*]

XSR#copy tftp: [[[//*location*]/*directory*]/*filename*]{*flash*: *cflash*:}[*filename*]

XSR#copy running-config startup-config

running-config	Keyword alias for current running configuration. This alias is only valid as follows: **copy running-config startup-config** This generates the current running configuration and saves it to **flash:startup-config.**
startup-config	Keyword alias for **flash:startup-config.**
flash:/cflash:	Alias for Flash or CompactFlash memory as a source or destination.
tftp:	Alias for a Trivial File Transfer Protocol (TFTP) network server which can be used as a source or destination. The syntax for this alias is **tftp:[[// location]/directory/]filename** The location must be an IP address. Default: 0.0.0.0

.

Note: A TFTP file network transfer may be lengthy especially when loading a software image which may be 3 - 6 Mbytes. The CLI prints a character every few seconds to indicate a transfer in progress.

Mode

Privileged EXEC: **XSR#**

Examples

XSR#copy tftp://192.168.27.1/root/enterasys-sw flash:

Save Configuration to TFTP Server

Save the **startup-config** file on a TFTP server over the network. Enter:

XSR#copy startup-config tftp: [[//location]/directory]/filename]

Software Image Loading from a TFTP Server

This XSR 1800 Series example loads the XSR software image into a file in Flash memory. If **flash:** is full, you must first delete the existing image file or rename the new image **xsr1800.fls** so as to copy over the old image. Be sure that your TFTP server is *running* and you know its IP address before you issue the command. Entering the **ipconfig** command at a DOS prompt will reveal the TFTP server IP address.

XSR#copy tftp://192.168.1.100/XSR1800.FLS flash:

Respond to the following script as prompted:

```
Destination file name [XSR1800.FLS]:
Copy 'XSR1800.FLS'from server
as 'XSR1800.FLS' into Flash (y/n) ?y
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Download from server done
File size: 1856714 bytes
```

The image is copied to **flash:** and its checksum verified. Should the transfer fail, then the router is temporarily without valid software in Flash and the XSR should ***not*** be reloaded or powered-down. A new TFTP copy should be initiated. The CLI session which initiated the copy is blocked during TFTP loading.

Configuration Load

This example loads *startup-config* via the network from a TFTP server. The XSR does not load the configuration from the network automatically.

```
XSR#copy tftp:TFTP1/tftpfiles/tftpimage flash:startup-config
```

Save Running Configuration

To save configuration changes into non-volatile memory, the running configuration must be copied into startup configuration:

```
XSR#copy running-config startup-config
```

copy running-config startup-config

This command copies the running configuration to the startup configuration file which is stored in non-volatile memory. It initiates a script requiring confirmation of your intention.

Syntax

```
copy running-config startup-config
```

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#copy running-config startup-config
```

Sample Output

```
XSR#copy running-config startup-config
Copy 'running-config'
  as 'startup-config' into flash: device (y/n) ? y
Running-config saved to startup-config.
<186>Sep 23 16:02:08 10.10.10.20 CLI: Running-config saved to startup-config by
user admin
```

copy startup-config tftp

This command saves the startup configuration on a TFTP server via the network connection. It initiates a script requiring confirmation of your intention.

Syntax

```
copy startup-config tftp:[[[//location]/directory]/filename]
```

location	IP address of the TFTP server on the network.
directory	Name of the TFTP directory.
filename	Name of the TFTP file.

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#copy startup-config tftp://192.168.1.100/cfg.txt
```

Sample Output

```
XSR#copy startup-config tftp://192.168.1.100/abc.cfg
Copy 'startup-config' from Flash to server
as 'abc.cfg'(y/n) ? y

Upload to server done
File size: 2997 bytes
```

delete <file>

This command removes a file from the XSR file system. It initiates a script requiring confirmation of your intention.

Syntax

delete [flash: | cflash:] *filename*

flash:	Flash memory directory.
cflash:	CompactFlash memory directory.
filename	Name of the file to be deleted.

Mode

Privileged EXEC: **XSR#**

Sample Output

```
XSR#delete startup-config
Delete filename [startup-config] y
Delete flash:y? [confirm] n
Delete of flash aborted
```

dir

This commands lists files in the Flash or CompactFlash directory.

Syntax

XSR#dir [flash: | cflash:]

flash:	Flash memory directory.
cflash:	CompactFlash memory directory.

Mode

Privileged EXEC: **XSR#**

Default

flash: unless you change the default using the **cd** command.

Example

```
XSR#dir flash:
```

Sample Output

The following is sample output from an XSR 1800 Series router:

```
XSR#dir flash:

Listing Directory flash:

size          date              time        name
817496        SEP-17-2002       15:21:32    bootrom1_18.fls
3220453       SEP-17-2002       15:24:08    xsr1800.fls
976           SEP-23-2002       16:02:08    startup-config
308           SEP-17-2002       15:26:14    user.dat
572           SEP-23-2002       14:50:32    cert.dat
0             SEP-23-2002       14:24:56    leases.cfg
64            SEP-23-2002       14:50:30    dhcpd.cfg
0             SEP-23-2002       14:24:56   leases.cfg.bak

2,328,576 bytes free
6,381,568 bytes total
```

more

This command shows a file's contents in ASCII format by default or hexadecimal (binary) format.

Syntax

XSR#more [/ascii | /binary | flash: | cflash:]filename

/ascii	File read in flat ASCII text.
/binary	File read in Hexadecimal format.
flash:	File residing in the On-Board Flash directory.
cflash:	File residing in the CompactFlash directory.
filename	Name of the file to be displayed.

Mode

Privileged EXEC: **XSR#**

Default

- Format: ASCII
- Directory: current directory

Examples

```
XSR#more /ascii flash:startup-config
XSR#more flash:startup-config
```

Sample Output

In ASCII format (**/ascii**):

```
Controller t1 1/0
Clock source line primary
Framing esf
```

In Binary format (**/binary:**):

```
00000000 12345678 12345678 12345678 12345678
00000010 12345678 12345678 12345678 12345678
00000020 12345678 12345678 12345678 12345678
```

pwd

This command displays the current directory.

Syntax

```
XSR#pwd
```

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#pwd
XSR#flash:
```

reload

This command allows the XSR to be rebooted (warm) or restarted (cold) with the option of successfully uploading a new image (the primary Enterasys Operating System [EOS] file) or *falling back* to the secondary (existing) file stored in *Flash:* or *Cflash:* if an error is detected. EOS Fallback tests the primary EOS and if it is not found, or verification fails, or errors appear in the *startup-config* file, or if no message is received from the configured SNMP server, the secondary EOS file is retained. Also, you can reboot or restart the XSR immediately or on a delayed basis. The EOS test duration begins when the primary EOS starts booting up and is variable to account for your network conditions.

One requirement of EOS fallback is to name the *primary-file*, described in the following Syntax table. Because the EOS test verifies this file to be a *bootable* image, it will reject the **reload fallback** command if verification fails. At this point the XSR will return to the *secondary* EOS file

which is specified in the *flash:boot-config* file. Although you cannot configure the secondary EOS file, if you wish to rename it, use the **boot system** command. Be aware that if the *boot-config* file does not exist in the *flash:* directory, EOS fallback will seach for the default **xsr1200.fls**, **xsr1800.fls** or **xsr3000.fls** file first in *flash:*, then in *cflash:*, finally over the network (as specified in the *bootrom* using the Bootrom monitor mode commands **sn** or **np**).

When you reboot the router using **reload**, the newly loaded **startup-config** file is converted to the **running config** file. The command initiates a script requiring confirmation of your intention. Be aware that the reload command does not appear in **startup-config**.

For more information on how to use this command, refer to the *Chapter 2: Managing the XSR* in the *XSR User's Guide*.

Syntax

reload [in | at [*mmm | hh:mm***] | cancel | cold | warm | fallback] primary-file {cflash: | flash:} duration [***config* **| snmp [***ip-address***]]**

in	Reloads after a specified interval, expressed in *minutes* or *hours:minutes.*
at	Reloads at a particular time, expressed in *hours* and *minutes*.
cancel	Cancels a pending reload.
primary-file	The filename, including the device name (*flash:xsr1800.fls*, for example), and can include any other designation of up to 31 ASCII characters. For example: *flash:my_new_xsr1800.fls or cflash:8_12_04_xsr1800.fls.*
cflash: *flash:*	Reloads primary OS file from cflash: or flash: directory and tested for an interval you specify between 5 and 30 minues.
duration	Primary OS test period after reload, ranging from 5 to 30 minutes.
config	Fallback to secondary OS file if any syntax error is found in *startup-config*.
snmp	Fallback to secondary OS file if no SNMP message was received during test.
ip-address	SNMP manager IP address to be monitored for received messages. If no SNMP IP address is specified, *any* received SNMP message indicates SNMP communications are successful.
cold	XSR hardware is re-initialized with the SDRAM cleared and software rebooted. The start is slower since hardware diagnostics are performed.
warm	XSR hardware is re-initialized and software rebooted. The start is faster since hardware diagnostics are not performed during the reboot.
none	Lack of argument performs a warm start.

Defaults

- Warm start
- Primary-OS test: 10 minutes

Mode

Privileged EXEC: **XSR#**

Examples

The following example immediately cold restarts the XSR:

```
XSR#reload cold
```

The following example *warm* upgrades the new image from the primary OS file in the *flash:* directory and tests it for 15 minute with the *fallback* option set to the secondary OS file if a syntax error is found in the *startup-config* file:

```
XSR#reload warm fallback flash:xsr1800.fls 15 config
```

The following example *warm* reboots the XSR in *240* hours and *12* minutes:

```
XSR#reload in 240:12 cold
```

The following command upgrades the new image via SNMP using the proprietary MIBs *enterasys-image-validation-mib* and *enterasys-configuration-management-mib*. For a description of the three-step procedure to configure the MIBs, refer to the *XSR User Guide*.

```
XSR#reload fallback cflash:xsr3004.fls 6 snmp 1.1.1.2
```

The following example upgrades the new image in 12 hours, 12 minutes with a fallback to the secondary OS if syntax errors are detected or if no SNMP messages are received from SNMP server at *192.168.57.4* during the test:

```
XSR#reload at 12:12 fallback config 10 config snmp 192.168.57.4
```

Sample Output

The following output is displayed, prompting you for a response, when you issue a *cold* reload:

```
XSR#reload cold
Proceed with reload (y/n)? y

X-Pedition Security Router Bootrom
Copyright 2004 Enterasys Networks Inc
....etc. proceeds with warm start
```

The following output is displayed when you *cancel* a reload:

```
XSR#reload cancel
No EOS Fallback is enabled
No reload is scheduled
```

rename

This command renames a file in the **Flash:** or **CFlash:** directory.

Syntax

```
rename {cflash: | flash:} source-filename destination-filename
```

cflash:	Renames a file within the CFlash: directory.
flash:	Renames a file within the Flash: directory
source-name	Source file name.
destination-name	Destination file name.

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#rename cflash:xsr3000.fls.5512 flash:xsr3000.fls
```

show hostname

This command displays the name you specified for the XSR.

Syntax

```
show hostname
```

Mode

EXEC: **XSR>**

Example

```
XSR#show hostname
```

Sample Output

```
XSR#show hostname
   Local hostname is XSR
```

show reload

This command displays data about scheduled reloads of the Enterasys Operating System (EOS).

Syntax

```
show reload
```

Mode

Privileged EXEC: **XSR#**

Sample Output

The following is sample output from the command when a reload is scheduled:

```
XSR#show reload
Reload scheduled in 9:56 minutes
   eos fallback                          running
   eos fallback                          not polling
   eos fallback crash monitoring         enabled
   eos fallback config                   disabled
   eos fallback snmp monitoring          enabled      192.168.72.72
   eos fallback test duration            5 minutes
   eos fallback primary file             flash:vpn_xsr1800.fls
   eos fallback is supported by installed bootrom 3.4 (need 3.4 or newer)
```

The following is sample output from the command when a reload is *not* scheduled:

```
XSR#show reload
No reload is scheduled
No EOS fallback
```

Parameter Description

running/not polling	Scheduled reload timer is running or the test period is in progress.
crash monitoring	A reload check for system failure.
fallback config	Fallback enabled or disabled.
snmp monitoring	A reload check for SNMP messages and SNMP server IP address.
test duration	The interval reload monitors for primary EOS crashes, a syntax error in *startup-config*, and SNMP messages for a configurable period between 5 and 30 minutes.
primary file	Directory and filename (including device name) of primary EOS file.

show running-config

This command displays the router's running configuration as a sequence of CLI commands segmented by module. The XSR gathers data from all system modules but collects and displays only those values different from default settings.

Syntax

```
show running-config
```

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#show running-config
```

Sample Output

The XSR 1800 Series sample output below displays as a number of CLI commands under the appropriate modules:

```
XSRtop(config)#show running-config

!PLATFORM
! CLI version 1.5
! XSR-1800
! Software:
!   Version 5.5.1.2, Built Jul 17 2003, 13:50:37

hostname XSRtop

!NETWORK MANAGEMENT
username admin privilege 15 "password is not displayed"
```

```
session-timeout console 35000
session-timeout telnet 35000
session-timeout ssh 35000

!T1E1
controller t1 0/2/0
clock source internal
no shutdown

!IKE
crypto isakmp proposal try1
authentication pre-share
encryption aes
hash md5
group 5
lifetime 40000
crypto isakmp peer 2.2.2.2 255.255.255.255
crypto isakmp peer 1.1.1.1 255.255.255.255

!IPSEC
crypto ipsec transform-set jj
no set security-association lifetime kilobytes
no set security-association lifetime seconds

!INTERFACE AND SUB-INTERFACE
interface FastEthernet 1
ip address 20.1.1.1 255.255.255.0
no shutdown

interface FastEthernet 2
ip address 1.1.1.16 255.255.255.0

interface Loopback5

int Dialer3

interface Serial 2/0:0
encapsulation ppp
ip address 30.1.1.1 255.255.255.0
no shutdown

interface Multilink 8
interface Vpn1 multi-point
interface Vpn4 point-to-point

!IP
ip local pool classA 10.10.0.0 255.255.0.0
ip route 1.1.1.0 255.255.255.0 2.2.2.2
ip route 7.0.0.0 255.0.0.0 Null0
```

```
!OSPF
router ospf 1
network 30.1.1.0 0.0.0.255 area 0.0.0.0
network 20.1.1.0 0.0.0.255 area 0.0.0.0

!RIP
router rip

!SNMP
snmp-server community public rw
snmp-server enable

!AAA
aaa group ii
dns server primary 0.0.0.0
dns server secondary 0.0.0.0
wins server primary 0.0.0.0
wins server secondary 0.0.0.0
pptp encrypt mppe 128
policy vpn
!
aaa method radius RADIUS
backup Radbackup
enable
group DEFAULT
address ip-address 0.0.0.0
hash enable
key 48aifij4
client firewall
auth-port 851
acct-port 850
attempts 5
retransmit 5
timeout 25
qtimeout 800

!FIREWALL
ip firewall network private 1.0.0.0 150.255.255.255 internal
ip firewall network any_ext 150.0.0.0 223.255.255.255 internal
ip firewall network allowRADIUS 10.10.10.1 mask 255.255.255.255 internal
ip firewall network allowRADIUS1 10.10.10.2 mask 255.255.255.255 internal
ip firewall network OSPFm 224.0.0.5 224.0.0.6 internal
ip firewall network Ten 10.1.0.0 mask 255.255.0.0 internal
!
ip firewall policy RADIUS allowRADIUS allowRADIUS1 Radius allow bidirectional
!
ip firewall filter OSPFm private Ten protocol-id 89
ip firewall filter OSPFm1 Ten private protocol-id 89 bidirectional
ip firewall load
```

verify

This command verifies a packed software image file. The file name must end in `*.fls`. If the directory name is not specified, the current directory is used.

Syntax

XSR#verify [flash: | cflash:]filename.fls

flash:	File located in the Flash directory.
cflash:	File located in the CompactFlash directory.
filename.fls	Name of a packed software image file.

Mode

Privileged EXEC: **XSR#**

Sample Output

The following sample XSR 1800 Series output displays a correct message:

```
XSR#verify xsr1800.fls
Verifying SW image file, j.fls
File chksum=0xeb14
SW Image size=070452 sum=0x6a9e compressed_size=1578677 entry=0x10000
Diagnostics size=815012 sum=0x2a32 compressed_size=266244 entry=0x10000

xsr1800.fls is a valid S/W image file
```

or an error message:

```
Invalid chksum(0xf2d9)!=Expected chksum0x4800
```

write

This command writes the running configuration to Flash memory, a network TFTP server, or a terminal. Only values different than default settings are collected and displayed.

Syntax

```
write erase
write terminal
write network flash: filename
write network tftp:[[/location]/directory/]filename
```

Sample Output

```
Controller t1 1/0
Clock source line primary
Framing esf\
etc.
```

Bootrom Monitor Mode Commands

Bootrom monitor mode offers special user access for Flash:/CompactFlash: file operations and on occasions when the XSR lacks valid software or runs abnormally. Enter the mode by pressing the key combination (**CTRL-C**) during the first five seconds of initialization. After you access the mode, list command groups by typing **h** to show the text below:

b	**Boot**
f	**Files**
n	**Network**
s	**Status**
t	**Time and Date**
D	**For Development Only**

All sub-commands in each group can be listed by entering the command group letter. The main menu provides the following functions:

- Reboot warm or cold
- Update Bootrom
- File system-related commands for the Flash ROM file system
- Modify network parameters
- Various status/show commands
  - Version number
  - Hardware information
  - Display crash information
- Display or change date and time on real-time clock
- Commands for development use only

bc

This command initiates a *cold* reboot.

bw

This command initiates a *warm* reboot.

bp

This command changes the Bootrom password. The default password is blank. You are prompted to enter a password by the following script:

```
XSR-1800: bp
Enter current password:
Enter new password: ******
Re-enter new password: ******
Password has changed.
```

If the Bootrom password is lost on the XSR 1800 Series, you can restore it by pressing the Default button. Be aware that when pressed, the Default button *erases all configuration files* and the master encryption key.

bu

This command updates the Bootrom file from a local file. You are prompted to enter data by the following script. When the "**Proceed with erasing current Bootrom in flash** ..." statement appears, enter **y**. Be sure not to interrupt the process or power down the XSR or it may be affected adversely. After you have updated this file, you can delete it from Flash to conserve space for other files. The following example displays output from an XSR 1800 Series router:

```
XSR-1800: bu cflash:bootrom1_20.fls
Checking cflash:bootrom1_18.fls...
Updating bootrom with file, "cflash:bootrom1_18.fls".
Proceed with erasing current Bootrom in flash and replace with
cflash:bootrom2_02.fls?y

**************************************************
*   Do not interrupt or power down until complete!  *
**************************************************

Erasing 8 sectors at address=0xfff00000
Programming 130816(0x1ff00) bytes at address 0xfff00100
Programming 131072(0x20000) bytes at address 0xfff20000
Programming 131072(0x20000) bytes at address 0xfff40000
Programming 131072(0x20000) bytes at address 0xfff60000
Programming 131072(0x20000) bytes at address 0xfff80000
Programming 131072(0x20000) bytes at address 0xfffa0000
Programming 31320(0x7a58) bytes at address 0xfffc0000
Programming high branch instruction at address 0xfffffffc
Verifying Bootrom flash sectors
Locking 8 Bootrom flash sectors

*****   Bootrom update completed.   *****

Using default Bootrom password. The system is not secure!!!
Use "bp" to change password
```

bU

This command updates the bootrom file through a network transfer to a local file. Be sure to enter an uppercase **U**. After you have updated this file, you can delete it from Flash to conserve space for other files.

cd

This command changes the current directory in the file system to **flash:** or **cflash:**.

copy

This command copies a file using the syntax **copy <source name> <destination name>**. You can copy from *flash:* to *cflash:* and vice versa.

da

This command displays system date and time with this sample output:

```
XSR-1800: da
Date: Thursday, 29-MAY-2003.
Time: 10:14:07
```

del

This command removes a file from **flash:** or **cflash:** memory.

df

This command displays free disk space with this sample output:

```
XSR-1800: df
Free space on flash: is 3383296 bytes (0x33a000).
```

dir

This command lists the contents of the current directory in long format. The XSR 1800 Series sample output is shown as follows:

```
XSR-1800: dir

size          date       time       name
--------       ------     ------    --------
 1728458    MAY-08-2002  03:05:14   xsr1800.fls
    1569    MAY-14-2002  02:25:00   startup-config
     214    JAN-01-2000  22:05:22   user.dat
  794828    JAN-01-2000  00:01:52   bootrom1_11.fls
       0    DEC-27-2019  11:07:14   cert.dat
    1352    JAN-18-2020  16:21:36   diagmsg.dat
  808220    MAY-08-2002  03:03:22   bootrom1_15.fls

    3383296(0x33a000) bytes free on flash:
```

The XSR 3000 Series sample output is shown as follows:

```
XSR-3250: dir

Listing Directory flash::
-rwxrwxrwx  1 0       0          4678118 May  5 23:06 xsr3000.fls
-rwxrwxrwx  1 0       0             2228 May 29 09:57 persistent-data
-rwxrwxrwx  1 0       0             1153 May 29 09:51 startup-config
-rwxrwxrwx  1 0       0                0 May 29 09:51 private-config

    2895872(0x2c3000) bytes free on flash:
```

ds

This command sets the system date using the syntax **yyyy mm dd w (1=Sunday)**. For example:

```
XSR-3020: ds 2003 6 1 3
```

dt

This command sets system time using the syntax **hh mm ss**. For example:

```
XSR:dt 11 59 59
```

ff

This command formats the Flash file system. We recommend you first save any **.dat**,**.cert**,**.cfg**, and your **startup-config** files to **cflash:** or a PC since any files in **flash:** will be deleted. You are prompted to enter data by the following script:

```
XSR-1800: ff
You will lose all files in the "flash:" file system.
       Are you sure you want to format the "flash:" file system? (y/n) y
Unlocking flash file sectors
Initializing DOS file system.
Formatting flashrom file system
.................................................... Done.
Set working directory to flash:

Using default Bootrom password. The system is not secure!!!
Use "bp" to change password
XSR-1800:
```

ffc

This command formats the CompactFlash file system.

ng

This command retrieves a file over the network using a remote IP address and remote file path.

np

This command modifies network parameters. You are prompted to enter data by the following script. While most of the options are self-explanatory, three require further description.

- When set to *no*, the *Autoboot* option places the prompt in Bootrom mode when you boot or power up the XSR.
- When set to *yes*, the default *Quickboot* action of delaying five seconds at startup for you to optionally enter **CTRL-C** and acquire Bootrom mode is negated. You can still acquire Bootrom mode, but you must immediately press **CTRL-C** upon seeing the *X-Pedition Security Router Bootrom* header.
- The default hostname (local target name), *XSR-1800*, cannot be changed. In the absence of a user-supplied hostname via the **hostname** CLI command, this name will be used as the CLI prompt and SNMP hostname in MIB-II.

```
XSR-1800: np

   Enter '.' = clear a field; '-' = go to previous field;  ^C = quit

Local IP address (192.168.1.1) :
Gateway IP address () :
```

```
Remote Host IP address (192.168.1.10) :
Remote file path (c:\) :
Use TFTP (no) :
Ftp userid (anonymous) :
Ftp password () :
Local target name (robo1) :
Autoboot (yes) :
Quick boot (no) :

Permanently save the network parameters? (y/n)
```

ns

This command saves a file over the network using a remote IP address/file path.

rename

This command renames a file using the syntax **rename <source name> <destination name>**

sb

This command displays boot parameters with this sample output:

```
XSR-1800: sb

Current boot file is xsr1800.fls
Boot selector default is flashrom, compactFlash, network
Available Network boot devices:   Eth1
```

sf

This command displays a fault report with the following sample output for the XSR 1800 Series. On XSR 3000 Series routers, you can enter **sf 0** or **sf 1** to display output from either CPU.

```
XSR-1800: sf
No fault report at 0x1feef00
```

This command displays the following sample output on the XSR-3250:

```
XSR-3250: sf

Software Revision: 6.0.0.0 without VPN; without Firewall
Creation Date:     Sep  7 2003, 16:07:42
Broadcom BCM1250 Rev 2 CPU0 up-time 0 hours 2 minutes 20 seconds
Crashed Task = PP, Task Status = 0, errno=0 initStage=0
Exception Vector Number=0x5, Address error exception, store
pc=      821014b0  sp=     85febb90  STATUS=  3400ff81
zero=    00000000  at=      08110000  v0=      11223344   v1=      00000000
a0=      3400ff81  a1=      00000000  a2=      3400ff81   a3=      85feb8f8
t0=      00000000  t1=      3400ff80  t2=      3400ff81   t3=      00000000
t4=      00000001  t5=      0000009b  t6=      0a0122d4   t7=      00000004
s0=      85febbe0  s1=      8219d3dc  s2=      00000000   s3=      00000000
s4=      00000000  s5=      00000000  s6=      00000000   s7=      00000000
t8=      00000000  t9=      00080000  k0=      eeeeeeee   k1=      00000000
```

```
gp=      8219b1e0  sp=      85febb90  s8=      00000000  ra=      820e9178
par1=    ffffffff  par2=    85febaf8  par3=    ffffffff  par4=    820e9b10
cause=   80000014  cntxt=   ffffffff  fpcsr=   d3800000  badva=   08112233
divLo=   00000000  divHi=   00000000  causeR=  ffffffff  fpcsr=   820e9170
BadVAddr=08112233

PP - Crashed Task Stack (sp=85febb90):
0x85feb790    ffffffff  00000000  00000008  ffffffff
0x85feb7a0    00000000  00000001  00000000  00000001
0x85feb7b0    00000000  8214ab00  0000000a  82142ee0
0x85feb7c0    ffffffff  85feb7c0  ffffffff  bf3285a4
0x85feb7d0    00000000  00000002  ffffffff  85feb7e0
0x85feb7e0    ffffffff  82154b50  00000000  00000017

......
```

si

This command displays XSR 1800 Series inventory with this sample output:

```
XSR-1800: si

          IBM PowerPC 405GP Rev. D
          Processor speed   = 200 MHz
          PLB speed         = 100 MHz
          OPB speed         = 33 MHz
          Ext Bus speed     = 25 MHz
          PCI Bus speed     = 33 MHz (Sync)
          Internal PCI arbiter enabled

          RAM installed: 64MB
          Flash installed: 8MB on processor board
          CompactFlash: SunDisk SDP 5/3 0.6 has 32047104 bytes
          Real Time Clock
          FastEthernet 1
          FastEthernet 2 Rev 0
          H/W Encryption Accelerator Rev 1
          T1E1 has 4 channelized ports on NIM slot 1. Rev 0
          ISDN BRI has 2 ST ports in NIM slot 2. Rev 1
          Empty internal NIM slot 3


          System up for 1500 seconds.
```

This command displays XSR 3000 Series inventory with this sample output:

```
XSR-3150: si
Hardware:
   Motherboard Information:
    XSR-3250 ID: 9002914-04 REV0A CPLD Rev 3
     Serial Number: 2914024201123206
     Processor: Broadcom BCM1250 Rev 2 at 600MHz
     PowerSupply1, PowerSupply2
    Fans 1 2 3 4 5 7 8 10
     CPU Temperature Max: 80C  Current: 35C
     Router Temperature Max: 60C  Current: 23C
```

```
 RAM: 512MB without interleave
    Memory Bus at 120MHz, CASL at 2.0
Bootrom Flash: 4MB
Filesystem Flash: 8MB
CompactFlash not present
 Real Time Clock
I/O on Motherboard:
GigabitEthernet  1 2 3
 Encryption Hardware: not present
 Slot 0 card 1: Empty
 Slot 0 card 2: Empty

System up for 9 days, 3 hours, 4 minutes 10 seconds.
```

sn

This command displays sample XSR 1800 Series network values:

```
XSR-1800: sn

Local IP address   : 10.120.112.33
Gateway IP address : 10.120.112.1
Remote IP address  : 10.120.112.88
Remote file path   : c:/tftpDir
Transfer Protocol  : TFTP
Local target name  : XSR1
Autoboot           : enabled
Quick boot         : no
IP address          : 192.168.1.1

       Current FastEthernet 0 MAC address is: 00:01:f4:01:01:01
       Current FastEthernet 1 MAC address is: 00:01:f4:01:01:02
```

sv

This command displays sample XSR 1800 Series bootrom version values:

```
XSR-1800: sv

               X-Pedition Security Router Bootrom
              Copyright 2003 Enterasys Networks Inc.

HW Version: 9002854-02 REV0A Serial Number: 0001F4000102
CPU: IBM PowerPC 405GP Rev. D
VxWorks version: 5.4
Bootrom version: 1.18
Creation date: Apr 14 2003, 10:12:36
```

4

Configuring Hardware Controllers

Observing Syntax and Conventions

The CLI command syntax and conventions use the notation described in the following table.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[*x*]	[ ] Square brackets indicate an optional parameter (*italic*)
[*x* | *y* | *z*]	[|] Square brackets with vertical bar indicate a choice of values
{*x* | *y* | *z*}	{ | } Braces with vertical bar indicate a choice of a required value
[*x* {*y* | *z*}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<*xx*>)	*xx* signifies interface type and number, e.g.: **F1**, **S2/1.0**, **D1**, **M57**, **G3. F** indicates a FastEthernet, and **G** a GigabitEthernet interface.
Next Mode entries display the CLI prompt after a command is entered.	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

Hardware Controller Commands

The following command sets allow you to define synchronization features for the XSR:

- "Hardware Controller Commands" on page 4-83
- "Hardware Controller Clear and Show Commands" on page 4-92

clock rate

This command configures the clock rate for the hardware connections on a serial interface. The command is valid and takes effect only when the interface is running in Asynchronous mode. For Synchronous mode, the clock rate is received externally.

Note: The clock rate cannot be changed in loopback mode.

Syntax

clock rate *bps*

bps	Configures the clock rate in bits per second (baud) on the line (async only). Valid rates are: 2400, 4800, 7200, 9600, 14400, 19200, 28800, 38400, 57600, and 115200.

Syntax of the "no" Form

no clock rate

Mode

Interface configuration: **XSR(config-if<S*x*>)#**

Default

9600

Example

XSR(config-if<S1/0>)#clock rate 19200

databits

This command sets the number of data bits accepted on a serial port. The command is valid and takes effect only when the interface is running in Async mode. In Sync mode, the clock rate is received externally.

Syntax

databits *bits*

bits	Number of databits per character on a serial port, ranging from 5 to 8.

Mode

Interface configuration: **XSR(config-if<S*x*>)#**

Syntax of the "no" Form

no databits

Default

8

Example

XSR(config-if<S1/0>)#databits 7

description

This command sets the description text for an interface. The description will appear in the *ifDescription* (interface description) variable of the MIB.

Syntax

description <*text*>

text	Alphanumeric characters which describe the interface.

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Syntax of the "no" Form

The *no* form of this command clears the description:

XSR(config-if<S1/0>)#no description

Example

XSR(config-if<S1/0>)#description "My FastEthernet Interface"

duplex

This command, used in conjunction with the **speed** command, forces the FastEthernet/ GigabitEthernet interface to operate at a specific duplex mode and speed. Setting the speed or duplex to auto-negotiate implies that both the speed and the duplex mode will be negotiated. It is not possible to manually set one and auto-negotiate the other. For example, you cannot set the speed to 10 Mb/s and set the duplex to auto-negotiate.

When issuing this command, be aware of the following additional conditions:

- Duplex mode *cannot* be changed while in loopback.
- Changing the duplex mode *preserves* the speed.
- When the speed is changed from **auto**, duplex will be set to **half**.
- Setting speed or duplex to **auto**, no speed, or no duplex sets both duplex and speed to auto.
- When connecting an **auto** setting on an XSR to a forced setting on another router, the forced setting *must* be set to **half-duplex** regardless of the speed (10 or 100 Mbits).
- When the Gigabit Fiber port is uses, both duplex and speed *must* be set to auto on *both* ends of the line to avoid an unpredictable link.

Syntax

duplex {*full* | *half* | *auto*}

full	Forces the interface to operate at full-duplex.
half	Forces the interface to operate at half-duplex.
auto	Allows the port to set the speed and duplex mode automatically.

Syntax of the “no” Form

```
no duplex
```

Default

auto

Mode

Interface configuration: **XSR(config-if<Fx>)#**

Example

```
XSR(config-if<F1/0>)#duplex full
XSR(config-if<F1/0>)#speed 100
```

loopback

This command forces the port into internal loopback mode. That is, the sender is internally connected to the receiver. This command is normallyused for diagnostic purposes only.

Note: Issuing this command will isolate the port from any connected network.

Syntax

```
loopback
```

Syntax of the “no” Form

```
no loopback
```

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

Off

Example

The following example resets interface *FastEthernet 1* to loopback:

```
XSR(config-if<F1>)#loopback
```

media-type

This command sets the media-type appropriate to the cable type that the interface is connected to.

Syntax

media-type {RS232 | RS422 | RS449 | RS530A | V35 | X21}

Note: The XSR Serial NIM does not detect the media-type of an attached cable. You must configure the correct interface media-type matching the attached cable for the serial interface to function properly.

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

RS232

Example

```
XSR(config-if<S1/0>)#media-type V35
```

nrzi-encoding

This command sets the encoding type to NRZI. It is valid and takes effect only when the interface is running in Synchronous mode. Some computers require the encoding type to be set to NRZI.

Syntax

```
nrzi-encoding
```

Syntax of the “no” Form

The *no* form of this command disable NRZI encoding:

```
no nrzi-encoding
```

Mode

Interface configuration: **XSR(config-if<S*x*>)#**

Default

Disabled

Example

```
XSR(config-if<S1/0>)#nrzi-encoding
```

parity

This command configures the parity on a serial interface. It is valid and takes effect only when the interface is in Asynchronous mode.

Syntax

parity {*even* | *mark* | *none* | *odd* | *space*}

even	Even parity.
mark	A constant *1* in the parity bit.
none	No parity.
odd	Odd parity.
space	A constant *0* in the parity bit.

Syntax of the "no" Form

The *no* form of this command invokes the *none* value:

no parity

Mode

Interface configuration: **XSR(config-if<Sx>)#**

Default

None

Example

```
XSR(config-if<S1/0>)#parity odd
```

physical-layer

This command specifies the mode of a serial interface as either synchronous or asynchronous. If set to *synchronous*, the port is configured as a DTE requiring an external transmit and receive clock to be supplied. If set to *asynchronous*, the interface will supply its own clock.

Note: A serial interface configured as a synchronous serial port *must* have an external transmit and receive clock.

Syntax

physical-layer {*sync* | *async*}

sync	Synchronous mode of XSR's serial interface.
async	Asynchronous mode of XSR's serial interface.

Mode

Interface configuration: **XSR(config-if<Sx>)#**

Default

Sync

Example

```
XSR(config-if<S1/0>)#physical-layer async
```

shutdown

This command disables an interface. When the interface is created, it is disabled by default.

Note: Issuing this command causes the interface to drop its link while disabled.

Syntax

```
shutdown
```

Syntax of the “no” Form

```
no shutdown
```

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

When the interface is created, it is disabled by default.

Example

```
XSR(config-if<S1/0>)#no shutdown
```

speed

This command, used in conjunction with the **duplex** command, forces the FastEthernet interface to operate at a specific speed and/or duplex mode. Setting the speed or duplex to auto-negotiate implies that both the speed and the duplex mode will be negotiated. It is not possible to manually set one and auto-negotiate the other. For example, you cannot set the speed to 10 Mb/s and set the duplex to auto-negotiate.

For GigabitEthernet only, to set 1000 Mbits speed for copper or fiber, select **auto** which will autosense the correct line and duplex speeds.

Keep in mind the following caveats:

- Changing the speed *preserves* the current duplex mode.

- Speed *cannot* be changed in loopback mode.
- When connecting an **auto** setting on an XSR to a forced setting on another router, the forced setting *must* be set to **half-duplex** regardless of the speed (10 or 100 Mbits).
- For GigabitEthernet only, you *must* use a cross-over cable when one or both ends of a line are forced. If both ends of the line are **auto** then you may use a cross-over *or* straight-through cable.
- When the Gigabit Fiber port is in use, both duplex and speed *must* be set to auto on *both* ends of the line otherwise the connection is unpredictable.

Syntax

speed {*10* | *100* | *auto*}

10	Forces the interface to operate at 10 Mbits per second.
100	Forces the interface to operate at 100 Mbits per second.
auto	Allows the port to set the speed and duplex mode automatically.

Syntax of the “no” Form

no speed

Mode

Interface configuration: **XSR(config-if<Fx>)#**

Default

Auto

Example

```
XSR(config-if<S1/0>)#speed auto
XSR(config-if<S1/0>)#duplex auto
```

stopbits

This command sets the number of stop-bits on a serial port. It is valid and takes effect only when the interface is running in asynchronous mode.

Syntax

stopbits {*1* | *2*}

1	One stop bit.
2	Two stop bits.

Syntax of the “no” Form

no stopbits

Mode

Interface configuration: **XSR(config-if<S*x*>)#**

Default

1

Example

The following example sets *2* stopbits on Serial port *1/0*:

```
XSR(config-if<S1/0>)#stopbits 2
```

vlan

This command configures a Virtual LAN (VLAN) ID on a sub-interface.

Note: Similar to the PPPoE sub-interface, you must issue the **no shutdown** command to keep the interface up.

Syntax

vlan *vlan-id*

vlan-id	Identifier of the sub-interface, ranging from 0 to 4094.

Syntax of the "no" Form

The *no* form of this command removes the VLAN ID configuration:

```
no vlan
```

Mode

Sub-Interface configuration: **XSR(config-if<*xx*>)#**

Examples

The following example configures a FastEthernet sub-interface with VLAN ID *10*:

```
XSR(config)#interface fastethernet 2.1
XSR(config-if<F2.1>)#vlan 10
XSR(config-if<F2.1>)#ip address 1.2.3.4 255.255.255.0
XSR(config-if<F2.1>)#no shutdown
```

The following example configures a VLAN configuration with PPPoE:

```
XSR(config)#interface fastethernet 2.4
XSR(config-if<F2.4>)#encapsulate ppp
XSR(config-if<F2.4>)#vlan 1400
XSR(config-if<F2.4>)#ip address negotiated
XSR(config-if<F2.4>)#ip mtu 1492
XSR(config-if<F2.4>)#no shutdown
```

Hardware Controller Clear and Show Commands

clear counters fastethernet

This command clears MIB-II counters for the FastEthernet interface. The counters cleared include:

- ifInOctets
- ifInUcastPkts
- ifInNUcastPkts
- ifInDiscards
- ifInErrors
- ifOutOctets
- ifOutUcastPkts
- ifOutNUcastPkts
- ifOutDiscards
- ifOutErrors
- ifInUnknownProtos

Syntax

clear counters fastethernet *interface sub-interface*

interface	FastEthernet interface number, ranging from 1 to 2.
sub-interface	FastEthernet sub-interface number, ranging from 1 to 64.

Mode

Privileged EXEC: **XSR#**

Example

The following example clears the MIB-II counters on FastEthernet port *1*, sub-interface *20*:

```
XSR#clear counters fastethernet 1.20
```

clear counters gigabitethernet

This command clears the same MIB-II counters for the interface as the **clear counters fastethernet** command.

Syntax

clear counters gigabitethernet *interface sub-interface*

interface	Interface number, ranging from 1 to 3.
sub-interface	Sub-interface number, ranging from 1 to 64.

Mode

Privileged EXEC: **XSR#**

Example

The following example clears the MIB-II counters on GigabitEthernet port *3*, sub-interface *2*:

```
XSR#clear counters gigabitethernet 3.2
```

clear interface fastethernet

This command resets the hardware logic on the FastEthernet interface. Using it preserves the current loopback mode, duplex mode and speed. This command is available on the XSR 1800 Series routers only.

Note: Issuing this command causes the interface to drop its link, any packets that it may have received, and any packets that may be in the process of being transmitted, while it resets. It *preserves* the current loopback mode, duplex mode and speed.

Syntax

clear interface fastethernet *number*

number	FastEthernet interface number ranging from 1 to 2.

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#clear interface fastethernet 2
```

clear interface gigabitethernet

This command resets the hardware on the GigabitEthernet interface. This command is available on the XSR 3000 Series routers only.

Note: Issuing this command causes the interface to drop its link, any packets that it may have received, and any packets that may be in the process of being transmitted, while it resets. It *preserves* the current loopback mode, duplex mode and speed.

Syntax

clear interface gigabitethernet *number*

number	GigabitEthernet port, ranging from 1 to 3, and sub-interface, ranging from 1 - 64.

Mode

Privileged EXEC: **XSR#**

Example

The following example resets GigabitEthernet port *1*, sub-interface *5*:

```
XSR#clear counters gigabitethernet 1.5
```

clear counters serial

This command clears serial interface counters. The counters cleared are:

- ifInOctets
- ifInUcastPkts
- ifInNUcastPkts
- ifInDiscards
- ifInErrors
- ifOutOctets
- ifOutUcastPkts
- ifOutNUcastPkts
- ifOutDiscards
- ifOutErrors
- ifInUnknownProtos

Syntax

clear counters serial [*card* / *port*]

card	XSR card number.
port	XSR port number.

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#clear counters serial 1/0
```

clear interface serial

This command resets the hardware logic on a serial interface.

Note: Issuing this command will cause the interface to drop its link, any packets that it may have received, and any packets that may be in the process of being transmitted, while it resets.

Syntax

clear interface serial [*card*/*port*]

card	XSR card number.
port	XSR port number.

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#clear interface serial 1/0
```

show controllers fastethernet

This command displays detailed FastEthernet controller data for a port. This interface is available on the XSR 1800 Series routers only.

Syntax

show controllers fastethernet ***number***

number	FastEthernet interface number, ranging from 1 to 2.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following example displays output from FastEthernet port 1:

```
XSR(config)#show controllers fastethernet 1
Packet Processor Tx Scheduler Stats:
        157 Packet driver Tx OK
        0 Packet driver not Tx: MUX END_ERR_BLOCK
        0 Packet driver not Tx: MUX ERROR
        0 Packet driver not Tx: Unknown Msg from MUX

The unit number is 1.
The interrupt number is 15.

Memory: base = 0xef600800
Vars: PollCount = 2806, g_eth1Interrupt = 0, bRxRunning = 0
Vars: bTxClean = 0, outQHung = 0
[...]

TX RING ENTRIES:
        The ring starts at 0x01fcd000.
        TxDRNum = 256, pTxMblkDR = 0x005f4824, TxDRIdx = 0
        TxDRCleanIdx = 0

        dataLen 0x00000000, status 0x00001300, buffer 0x00000000
```

```
        dataLen 0x00000000, status 0x00001300, buffer 0x00000000
        dataLen 0x00000000, status 0x00001300, buffer 0x00000000
        dataLen 0x00000000, status 0x00001300, buffer 0x00000000
        dataLen 0x00000000, status 0x00001300, buffer 0x00000000
        [...]

RX RING ENTRIES:
        The ring starts at 0x01fcc000.
        RxDRNum = 128, pRxMblkDR = 0x01f33c88, RxDRIdx = 19
        RxBuffSize = 1728, RxBuffOffset = 160

        dataLen 0x00000000, status 0x00008000, buffer 0x01cc6c20
        dataLen 0x00000000, status 0x00008000, buffer 0x01cc72e0
        dataLen 0x00000000, status 0x00008000, buffer 0x01cc79a0
        dataLen 0x00000000, status 0x00008000, buffer 0x01cc8060
        dataLen 0x00000000, status 0x00008000, buffer 0x01cc8720
        [...]
```

show controllers gigabitethernet

This command displays detailed FastEthernet controller data for an interface. This command is available on the XSR 3000 Series routers only.

Syntax

show controllers gigabitethernet [*number*]

number	GigabitEthernet interface, ranging from 1 to 3.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following example displays output from GigabitEthernet port *1*:

```
XSR#show controllers gigabitethernet 1
Packet Processor Tx Scheduler Stats:
        0 Packet driver Tx OK
        0 Packet driver not Tx: MUX END_ERR_BLOCK
        0 Packet driver not Tx: MUX ERROR
        0 Packet driver not Tx: Unknown Msg from MUX

The unit number is 1.
The interrupt number is 63. The source is 19.
The PHY is 1

Memory: base=0xb0064000

Vars: g_eth1Interrupt=0, mClBlkSize=0, bufsize=0

TX RING:
        Ring starts at 0x815b1620.
        TMaxDR=512, pTCurrDR=0x00000c30, TAddidx=0
        TRemidx=0
```

```
        datalen 0x00000000, status 0x00000000, buffer 0x80000000
        datalen 0x00000000, status 0x00000000, buffer 0x80000000
        datalen 0x00000000, status 0x00000000, buffer 0x80000000
        datalen 0x00000000, status 0x00000000, buffer 0x80000000
        datalen 0x00000000, status 0x00000000, buffer 0x80000000
        [...]

RX RING:
        Ring starts at 0x81568c60.
        RMaxDR=512, pRCurrDR=0x00000830, RIdx=0

        datalen 0x00000000, status 0x00000000, buffer 0x8ff5df60
        datalen 0x00000000, status 0x00000000, buffer 0x8ff5e620
        datalen 0x00000000, status 0x00000000, buffer 0x8fe86ce0
        datalen 0x00000000, status 0x00000000, buffer 0x8fe873a0
        datalen 0x00000000, status 0x00000000, buffer 0x8fe87a60
        [...]

The secondary MAC addresses are (in hex):
                [0] : < not used >
                [1] : < not used >
                [2] : < not used >
                [3] : < not used >
```

show controllers serial

This command displays detailed serial controller data for an interface.

Syntax

show controller serial *card/port*

card	XSR card number of the serial controller.
port	XSR port number of the serial controller.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following example displays output from Serial port *1*/0:

```
XSR#show controllers serial 1/0

Forward Engine Serial Layer Tx/Rx Stats:

RX FROM UPPER LAYER & TX TO DRIVER
        Pcks Rx         = 0
        Pcks Tx         = 0
        Pcks Discarded = 0
RX FROM DRIVER & TX TO UPPER LAYER
        Pcks Rx         = 0
        Pcks Tx         = 0
        Pcks Discarded = 0
```

```
Packet Processor Tx Scheduler Stats:
        0 Packet driver Tx OK
        0 Packet driver not Tx: MUX END_ERR_BLOCK
        0 Packet driver not Tx: MUX ERROR
        0 Packet driver not Tx: Unknown Msg from MUX

The unit number is 50331656.
The interrupt number is 26.
The DSR poll count is 800 ms.
The ACCM is at 0x01040acc.

Vars: CCR2=0x98ff0500, CCR1=0x98ff0500, CCR0=0x00000000, CD=0, g_Ser=0
Vars: bHandleRx=0, bTxClean=0
Vital Stats: TX Q Items = 0, TX Q Bytes = 0, TX CLK = 0

Memory: base = 0xa0020000


TX RING ENTRIES:
     The interrupt ring starts at 0x018d6b60 (IDX = 0).

     The data ring starts at 0x018f4d60.
     TpTxMblkDR = 0x0104055c, TxDRIdx = 1, TxDRCleanIdx = 1

(-2) next 0xa04d8f21, flag1 0x00000000, flag2 0x00000000, buffer 0x00000000
(-1) next 0xc04d8f21, flag1 0x00000000, flag2 0x00000000, buffer 0x00000000
( 0) next 0xe04d8f21, flag1 0x00000000, flag2 0x00000000, buffer 0x00000000
( 1) next 0x004e8f21, flag1 0x00000000, flag2 0x00000000, buffer 0x00000000
( 2) next 0x204e8f21, flag1 0x00000000, flag2 0x00000000, buffer 0x00000000
     [...]

RX RING ENTRIES:
     The interrupt ring starts at 0x018d6ac0 (IDX = 0).

     The data ring starts at 0x018f3540.
     RxDRNum = 64, pRxMblkDR = 0x018f6b8c, RxDRIdx = 0
     RxBuffSize = 1728, RxBuffOffset = 160

(-2) next 0x60358f21, flag1 0x0000fc05, flag2 0x00000000, buffer 0xe07d5021
(-1) next 0x80358f21, flag1 0x0000fc05, flag2 0x00000000, buffer 0xa0845021
( 0) next 0xa0358f21, flag1 0x0000fc05, flag2 0x00000000, buffer 0x608b5021
( 1) next 0xc0358f21, flag1 0x0000fc05, flag2 0x00000000, buffer 0x20925021
( 2) next 0xe0358f21, flag1 0x0000fc05, flag2 0x00000000, buffer 0xe0985021
     [...]
```

show interface bri

This command displays ISDN Basic Rate Interface (BRI) information for an interface.

Syntax

show interface bri [*card/port*:*channel.sub-interface*]

card	ISDN BRI card number, either 1 or 2.
port	ISDN BRI port number, either 0 or 1.

channel	ISDN BRI D- or B-channel, either 0 for the D-channel, and 1 or 2 for the B-channels.
sub-interface	ISDN BRI sub-interface, ranging from 1 to 30.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following example displays output by the command:

```
XSR(config)#show interface bri 1/0

********** Serial Interface Stats **********
D-Serial  1/0:0 is Admin Up / Oper Down

********************* ISDN  Stats ISDN-BRI  1/0 *********************
Layer 1: DOWN  Layer 2: DOWN  State: OFFLINE      Admin Up        Oper Down

Term. 1 Spid:2200555 State:  OFFLINE Cause: 000
Term. 2 Spid:2201555 State:  OFFLINE Cause: 000
Total Length = 257

The name of this device is bri0/1/0:0 .

The slot is 0.
The card is 1.
The port is 0.
The channel is 0.
The current MTU is 1500.
The device is in polling mode, and is active.
The channel is logically INACTIVE.
The operational state is OPER_DOWN.
The protocol used is LAPD.
The baud rate is 16000 bits/sec.
The device uses CRC-16 for Tx.
The device uses CRC-16 for Rx.

Other Interface Statistics:
        ifindex              0
        ifType               75
        ifAdminStatus        1
        ifOperStatus         2
        ifLastChange         00:00:00
        ifInOctets           0
        ifInUcastPkts        0
        ifInNUcastPkts       0
        ifInDiscards         0
        ifInErrors           0
        ifInUnknownProtos    0
        ifOutOctets          0
        ifOutUcastPkts       0
        ifOutNUcastPkts      0
        ifOutDiscards        0
        ifOutErrors          0
        ifOutQLen            16
```

show interface dialer

This command displays information about the Dialer interface.

Syntax

show interface dialer [*number*]

number	Dialer interface number, ranging from 0 to 255.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following example displays information about Dialer interface *3*:

```
XSR#show interface dialer 3

********** Dialer Interface Stats **********
Dialer3 is Admin Down
Internet address is not assigned

Dialer3
Dialer state is: DOWN
Wait for carrier default: 60, default retry: 3
Dial String     Success Failures        Map Class
Free pool ISDN channels: <0>
Free pool serial ports:  <0>
```

show interface fastethernet

This command displays information about a FastEthernet interface. This interface is available on the XSR 1800 Series routers only.

Syntax

show interface fastethernet [*number*]

number	FastEthernet interface number of 1 or 2.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following is sample output from FastEthernet interface *1*:

```
XSR#show interface FastEthernet 1
FastEthernet1 is Admin Up
Internet address is 51.51.51.1, subnet mask is 255.255.255.0
Internet address is 52.52.52.1, subnet mask is 255.255.255.0 Secondary
Internet address is 53.53.53.1, subnet mask is 255.255.255.0 Secondary
```

```
Internet address is 54.54.54.1, subnet mask is 255.255.255.0 Secondary
Internet address is 57.57.57.1, subnet mask is 255.255.255.0 Secondary
Internet address is 58.58.58.1, subnet mask is 255.255.255.0 Secondary

The name of this device is Eth1.

The physical link is currently up.
The device is in polling mode, and is active.
The last driver error is '(null)'.

The duplex mode is set to auto-negotiated.
The current operational duplex mode is negotiated to half.

The speed is set to auto-negotiated.
The current operational speed is negotiated to 100 Mb/s.

The MAC address is (in hex) 00:01:f4:0d:26:72.
The MTU is 1500.
The bandwidth is 100 Mb/s.

Other Interface Statistics:
        ifindex              0
        ifType               6
        ifAdminStatus        1
        ifOperStatus         1
        ifLastChange         00:32:39
        ifInOctets           529727
        ifInUcastPkts        0
        ifInNUcastPkts       7328
        ifInDiscards         0
        ifInErrors           0
        ifInUnknownProtos    0
        ifOutOctets          157800
        ifOutUcastPkts       0
        ifOutNUcastPkts      157
        ifOutDiscards        0
        ifOutErrors          0
        ifOutQLen            256
```

The following is sample output from a VLAN interface on FastEthernet sub-interface *2.1*:

```
XSR#show interface FastEthernet 2.1
FastEthernet2.1 is Admin Up
Internet address is 1.2.3.4, subnet mask is 255.255.255.0
Other Interface Statistics:
        ifOperStatus         1
        ifInOctets           956932
        ifOutOctets          495034
Configured VLANs:
        VLAN Id              1200
```

The following is sample output from a VLAN interface on FastEthernet sub-interface *2.4* configured with PPPoE:

```
XSR#show interface FastEthernet 2.4
FastEthernet2.4 is Admin Up
Internet address is 5.5.5.4, subnet mask is 255.255.255.0
LCP         State: OPENED
IPCP        State: OPENED

The logical link is currently Up
```

```
The Name of the Access Concentrator is c3600-1
The Session Id is 0x0005
The MAC Address of the Access Concentrator is  0x00:30:85:20:47:62
The MTU is 1492
Other Interface Statistics:
        ifOperStatus         1
        ifInOctets           119439
        ifOutOctets          119256
Configured VLANs:
        VLAN Id              1400
PPP Encapsulation
```

show interface gigabitethernet

This command displays information about a GigabitEthernet interface which is available on XSR 3000 Series routers only.

Syntax

show interface gigabitethernet [*number*]

number	The GigabitEthernet interface, ranging from 1 to 3, and sub-interface. Range: 1 to 64.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following example is sample output from GigabitEthernet interface 1:

```
XSR#show interface gigabitethernet 1
GigabitEthernet 1  is Admin Up
Internet address is 150.50.1.14, subnet mask is 255.255.255.0
The name of this device is Eth1.

The physical link is currently DOWN.
The active port is copper.
The device is in polling mode, and is active.
The last driver error is '(null)'.

The duplex mode is set to auto-negotiated.
The current operational duplex mode is not yet determined.

The speed is set to auto-negotiated.
The current operational speed is not yet determined.

The Primary MAC address is (in hex) 00:01:f4:2b:3e:1b.
The MTU is 1518.
The bandwidth is 10 Mb/s.

Other Interface Statistics:
        ifindex              0
        ifType               6
        ifAdminStatus        1
        ifOperStatus         2
```

```
ifLastChange          00:00:00
ifInOctets            0
ifInUcastPkts         0
ifInNUcastPkts        0
ifInDiscards          0
ifInErrors            0
ifInUnknownProtos     0
ifOutOctets           0
ifOutUcastPkts        0
ifOutNUcastPkts       0
ifOutDiscards         0
ifOutErrors           0
ifOutQLen             256
```

show interface loopback

This command displays information about the loopback interface.

Syntax

show interface loopback [*number*]

number	Loopback address number ranging from 0 to 15.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following is sample output from Loopback interface 5:

```
XSR#show interface loopback5
Loopback5 is Admin Up
Description: My loopback interface
Internet address is 57.57.57.57, subnet mask is 255.255.255.0
```

show interface multilink

This command displays information about the Multilink interface.

Syntax

show interface multilink [*number*]

number	Multilink address number, ranging from 1 to 32767.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following is sample output from Multilink interface *8*:

```
XSR#show interface multilink 8

********** Multilink Interface Stats **********
Multilink 8 is Admin Down
Internet address is not assigned
LCP       State: CLOSED
Multilink State: CLOSED

Max Fragment delay is 10 ms

MLPPP Bundle Info:
Control Object state is Admin Down / Oper Down
Multilink PPP has no memberlinks

Data Object state is Admin Down
The adjacent is DOWN and data passing is FALSE
Bundle size is 0
Max Load Threshold: 0
Total Load Bandwidth is 64000 bits/sec
Bundle Stats
Rx: Total          0, TX: Total          0
    Data           0,     Data           0
    Ctrl           0,     Ctrl           0
    Null           0,     Null           0
    Drop           0,     Drop           0
Rx Load BW Avg   0, Max   0, Min   0
Tx Load BW Avg   0, Max   0, Min   0
```

show interface null

This command displays attributes of the null interface (*Null 0*), an IP interface which uniquely does not require an IP address to appear. It is installed automatically by the XSR so that discard routes can be employed by OSPF. You cannot configure this interface, it is always administratively up and cannot be deleted.

The Null interface displays only when you enter **`show ip interface null 0`** or **`show interface null 0`**. If it is not *specified* in the **`show interface`** or **`show ip interface`** commands, it will not display. Also, it does not display in the *running-config* file.

Syntax

```
show interface null 0
```

Mode

Privileged EXEC or Global configuration: **`XSR#`** or **`XSR(config)#`**

Sample Output

The following example is sample output from the **`show ip interface null 0`** command:

```
XSR#show ip interface null 0
Null0 is Admin Up
Internet address is not assigned
Rcvd:  0 octets, 0 unicast packets,
       0 discards, 0 errors, 0 unknown protocol.
```

```
Sent:  0 octets, 0 unicast packets,
       0 discards, 0 errors.
MTU is 1500 bytes.
Proxy ARP is enabled.
Helper address is not set.
Directed broadcast is enabled.
Outgoing access list is not set.
Inbound access list is not set.
IP Policy Based Routing is not enabled.
```

The following example is sample output from the **show interface null 0** command:

```
XSR#show interface null 0
Null0 is Admin Up
Internet address is not assigned
```

show interface serial

This command displays general information for a serial interface.

Syntax

show interface serial [*card*/*port*]

card	XSR card number of serial interface.
port	XSR port number of serial interface.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following example displays output from Serial interface *1/0*:

```
XSR#show interface serial 1/0

********** Serial Interface Stats **********
Serial 1/0 is Admin Down / Oper Down
Internet address is 200.163.21.1

The name of this device is Ser1/0.

The card is 1.
The channel is 0.
The current MTU is 1500.
The device is in polling mode, and is ACTIVE.
The last driver error is (null).
The physical-layer is HDLC-SYNC.
The baud rate is estimated to be 1024000 bits/sec.
The device uses CRC-16 for Tx.
The device uses CRC-16 for Rx.
The type of encoding is NRZ.
The media-type is RS-232/V.28 (DTE).
The loopback mode is off.

Other Interface Statistics:
```

```
ifindex              0
ifType               22
ifAdminStatus        1
ifOperStatus         1
ifLastChange         00:00:25
ifInOctets           1500
ifInUcastPkts        100
ifInNUcastPkts       0
ifInDiscards         0
ifInErrors           0
ifInUnknownProtos    0
ifOutOctets          2134
ifOutUcastPkts       14
ifOutNUcastPkts      0
ifOutDiscards        0
ifOutErrors          0
ifOutQLen            280
```

show interface vpn

This command displays attributes of the configured VPN interface.

Syntax

show interface vpn [*0-255*]

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following is sample output displaying VPN interface *57* statistics:

```
XSRtop#show interface vpn 57
Vpn 57 is Admin Up
Internet address is 4.4.4.4, subnet mask is 255.255.255.0
Multicast redirect to 6.6.6.6 is enabled.
This interface includes the VPN tunnel 'Boston'.
The tunnel peer's Internet IP address is 0.0.0.0.
The tunnel encapsulation protocol is UNKNOWN.
The identity used to initiate the tunnel is ''
The tunnel's current state is Disabled.
```

5

Configuring the Internet Protocol

Observing Syntax and Conventions

The CLI Syntax and conventions use the notation described below.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[*x*]	[ ] Square brackets indicate an optional parameter (*italic*)
[*x* | *y* | *z*]	[|] Square brackets with vertical bar indicate a choice of values
{*x* | *y* | *z*}	{ | } Braces with vertical bar indicate a choice of a required value
[*x* {*y* | *z*}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<*xx*>)	*xx* signifies interface type and number, e.g.: **F1**, **S2/1.0**, **D1**, **M57**, **G3.F** indicates a FastEthernet, and **G** a GigabitEthernet interface.
Next Mode entries display the CLI prompt after a command is entered.	
Sub-command headings are displayed in ***red*** text.	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

IP Commands

The following command sets define IP functionality on the XSR including:

- "OSPF Commands" on page 5-84.
- "OSPF Debug and Show Commands" on page 5-104.
- "RIP Commands" on page 5-123.
- "RIP Show Commands" on page 5-136.
- "RTP Header Compression Commands" on page 5-137.
- "Policy-Based Routing Commands" on page 5-145.
- "PBR Clear and Show Commands" on page 5-148.
- "ARP Commands" on page 5-149.
- "Other IP Commands" on page 5-151.
- "IP Clear and Show Commands" on page 5-168.
- "Network Address Translation Commands" on page 5-182.
- "Virtual Router Redundancy Protocol Commands" on page 5-191.

- "VRRP Clear and Show Commands" on page 5-197.

OSPF Commands

area authentication

This command enables/disables authentication for an OSPF area.

Syntax

area *area-id* **authentication** [*message-digest*]

area-id	OSPF area to be authenticated, expressed in decimals or IP addresses.
message-digest	Enables MD5 authentication on the OSPF area indicated by *area-id* keyword

Syntax of the "no" Form

The *no* form of this command removes authentication from the OSPF area specified by *area-id*:

no area *area-id* **authentication**

Mode

Router configuration: **XSR(config-router)#**

Default

The default value is Type 0 authentication; that is, *no* authentication.

Example

This example enables authentication on OSPF area *10.0.0.0*. interface *Serial 1/1*, whose address is *172.16.77.1*, is part of area *10.0.0.0*, so an authentication mechanism could be defined for it:

```
XSR(config)#interface serial 1/1
XSR(config-if<S1/1)#ip address 172.16.77.1 255.255.255.0
XSR(config-if<S1/1)#ip ospf message-digest-key 20 md5 pass1
XSR(config)#router ospf 1
XSR(config-router)#network 172.16.77.1 0.0.0.0 area 10.0.0.0
XSR(config-router)#area 10.0.0.0 authentication message-digest
```

area default-cost

This command sets the cost value for the default route that is sent into a stub area by an Area Border Router (ABR). This command is restricted to ABRs attached to stub areas.

Syntax

area *area-id* **default-cost** *cost*

area-id	The stub area expressed in decimals or IP addresses.
cost	Cost value for a summary route that is sent to a stub area by default. Valid values are 24-bit numbers, from 0 to 16,777,215.

Syntax of the "no" Form

The *no* form of this command removes the *cost* value from the summary route that is sent by default into the stub area identified by the *area-id*:

no area *area-id* **default-cost**

Mode

Router configuration: **XSR(config-router)#**

Default

1

Example

The following command sets the cost value for the stub area *10* as *99*:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#ip address 172.16.101.5 255.255.255.252
XSR(config-if<S1/0>)#router ospf
XSR(config-router)#network 172.16.101.5 0.0.0.0 area 10
XSR(config-router)#area 10 stub no-summary
XSR(config-router)#area 10 default-cost 99
```

area nssa

This command configures an area as a Not So Stubby Area (NSSA) which allows some external routes represented by external Link-State Advertisements (LSAs) to be imported into it. This is in contrast to a stub area that does not allow any external routes. External routes that are not imported into an NSSA can be represented by means of a default route. It is used when an OSPF inter-network is connected to multiple non-OSPF routing domains.

Syntax

area *area-id* **nssa** [*default-information-originate*]

area-id	NSSA area expressed in decimals or IP addresses.
default-information-originate	Generates a default of Type 7 into the NSSA. It is used when the router is a NSSA ABR

Syntax of the "no" Form

The *no* form of this command changes the NSSA back to a plain area:

no area *area-id* **nssa** [*default-information-originate*]

Mode

Router configuration: **XSR(config-router)#**

Default

No NSSA defined

Example

The following example configures area *10* as a NSSA area:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip address 172.16.10.5 255.255.255.252
XSR(config)#router ospf 1
XSR(config-router)#network 172.16.10.5 0.0.0.0 area 10
XSR(config-router)#area 10 nssa default-information-originate
```

area range

This command defines the range of addresses to be used by Area Boundary Routers (ABRs) when they communicate routes to other areas. ABRs summarize an area's *intra*-area routes into *inter*-area routes which are then injected into other areas. The metric used is the *highest* metric of the included *intra*-area routes. The forwarding address is 0.

Other actions implemented by this command include:

- A summary range becomes active if it includes at least one *intra*-area route being leaked into the area.
- A discard route is installed for an active summary range. Conversely, when it becomes inactive, the discard route is removed.
- The cost of the summary range is the highest cost among all leaked *intra*-area routes.
- SNMP supports area range via MIB object *ospfAreaRangeTable* as defined in RFC-1850.

Note: You should avoid needless reorigination of Type-3 Link-State Advertisements (LSAs). For example, leaking intra-area routes which do not change the cost of a summary will re-origination the summary LSA.

Syntax

area *area-id* **range** *ip-address mask* [*advertise*][*not-advertise*]

area-id	Area at the boundary of which routes will be summarized. Valid values are decimals or IP addresses.
ip-address	Common prefix of summarized networks.
mask	Length of the common prefix.
advertise	Broadcasts a single Type-3 LSA for all intra-area routes leaked into this area and included in the summary range.
not-advertise	Suppresses Type-3 LSA generation for all routes in the summary range.

Syntax of the "no" Form

The *no* form of this command bars routes from being summarized:

no area *area-id* **range** *address mask*

Mode

Router configuration: **XSR(config-router)#**

Examples

This example sets the address range used by this router for summarized routes learned at the boundary of area *0.0.0.0*, as *172.16.0.0/16*:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip address 172.16.16.1 255.255.240.0
XSR(config)#router ospf 1
XSR(config-router)#network 172.16.16.1 0.0.0.0 area 0.0.0.0
XSR(config-router)#area range 0.0.0.0 172.16.0.0 255.255.0.0
```

The following example aggregates *64.64.64.0/24* in area *1* into summary route *64.0.0.0/8* and makes the summary available for creation of inter-area routes:

```
XSR(config)#router ospf 1
XSR(config-router)#area 1 range 64.0.0.0 255.0.0.0
```

area stub

This command defines an area as a stub area.

Syntax

area *area-id* **stub** [*no-summary*]

area-id	Stub area expressed in decimals or IP addresses.
no-summary	Bars an ABR from sending LSAs into the stub area. When used, this value means all destinations outside the stub area are represented via a default route.

Syntax of the “no” Form

The *no* form of this command changes the *stub* back to a plain area:

no area *area-id* **stub** [***no-summary***]

Mode

Router configuration: **XSR(config-router)#**

Defaults

Disabled

Example

The following example defines area *10* as a stub area:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip address 172.16.152.1 255.255.255.0
XSR(config-if<F1>)#exit
```

```
XSR(config)#router ospf
XSR(config)#network 172.16.152.0 0.0.0.0 area 10
XSR(config)#area 10 stub
```

area virtual-link

This command defines an OSPF virtual link, which represents a logical connection between the backbone and a non-backbone OSPF area. Backbones are areas including all ABRs, networks not wholly contained in any area, and their attached routers.

Syntax

area *area-id* **virtual-link** *router-id* [**authentication** [**message-digest** | **null**]] [**hello-interval** *seconds*] [**retransmit-interval** *seconds*] [***transmit-delay*** *seconds*] [**dead-interval** *seconds*] [**authentication-key** *key* | **message-digest**-*key keyid* **md5** *key*]

area-id	Transit area for the virtual link - expressed as decimal or IP addresses - and through which a virtual link is established.
router-id	The ABR's Router ID. A virtual link is built from the ABR, where virtual link configuration occurs. You can configure a loopback address for the XSR to be used as the Router ID with the **interface loopback** command. If no loopback address is defined, the Router ID is the highest non-zero IP address of existing configured and active interfaces.
authentication	Authentication type.
message-digest	MD5 authentication is used.
null	No authentication is used.
hello-interval *seconds*	Interval between hello packets on a port. It must be the same for all nodes attached to a network. Range: 1 to 3600 seconds.
retransmit-interval *seconds*	Interval between successive retransmissions of the same LSAs. Valid values are greater than the expected period for the update packet to reach and return from the port, ranging from 1 to 3600 seconds.
transmit-delay *seconds*	Estimated interval for a link state update packet on the port to be transmitted, ranging from 1 to 3600 seconds.
dead-interval *seconds*	Interval that hello packets of a router are not communicated to neighbor routers before the neighbor learn that the router sending the hello packet is out of service. This value must be the same for all nodes attached to a certain subnet, and ranges from 1 to 3600 seconds.
authentication *key*	Password used by neighbor routers. Valid values are alphanumeric strings up to 8 bytes. Neighbor routers on a network must have the same password.
message-digest *keyid* ***md5*** *key*	Specifies a key id and a password (key) for MD5 authentication. Neighbor routers and this router use the keyid and key. Valid values for *keyid* are 1 to 255. Valid values for the key are alphanumeric strings of up to 16 characters. Neighbor routers on a network must have the same *keyid* and *key*.

Syntax of the "no" Form

The *no* form of this command removes the *virtual link*:

no area *area-id* **virtual-link** *router-id* [**authentication** [**message-digest** | **null**]] [**hello-interval** *seconds*] [**retransmit-interval** *seconds*] [***transmit-delay*** *seconds*] [**dead-interval** *seconds*] [**authentication-key** *key* | **message-digest**-*key keyid* **md5** *key*]

Mode

Router configuration: **XSR(config-router)#**

Defaults

- hello-interval seconds: 10 seconds
- retransmit-interval seconds: 5 seconds
- transmit-delay seconds: 1 second
- dead-interval seconds: 40 seconds
- authentication-key key: No default
- message-digest-key keyid md5 key: No default

Example

The following example, as illustrated in Figure 5-1, shows the virtual link configuration for two ABRs. *ABR1* physically interfaces area 2 to the backbone (area 0.0.0.0). *ABR2* physically interfaces area 3 to area 2. A virtual link is created between the two ABRs by means of area 2, which becomes the transit area. The RouterID for ABR1 is *192.168.33.1*. The RouterID for ABR2 is *192.168.33.2*.

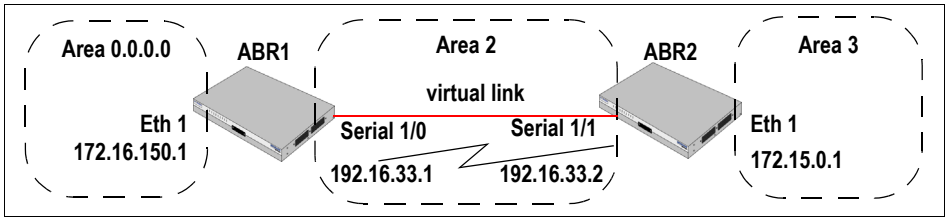
On *ABR1* enter the following commands:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip address 172.16.150.1 255.255.255.0
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#ip address 192.16.33.1 255.255.255.0
XSR(config)#router ospf 1
XSR(config-router)#network 172.16.150.0 0.0.0.255 area 0.0.0.0
XSR(config-router)#network 192.16.33.0 0.0.0.255 area 2
XSR(config-router)#area 2 virtual-link 192.16.33.2
```

On *ABR2* enter the following commands:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip address 172.15.0.1 255.255.0.0
XSR(config)#interface serial 1/1
XSR(config-if<S1/1)#ip address 192.16.33.2 255.255.255.0
XSR(config)#router ospf 1
XSR(config-router)#network 172.15.0.1 0.0.0.0 area 3
XSR(config-router)#network 192.16.33.0 0.0.0.255 area 2
XSR(config-router)#area 2 virtual-link 192.16.33.1
```

Figure 5-1 Area Virtual Link Example



auto-virtual-link

This command automatically creates virtual links. Refer to the **`area-virtual-link`** command for more related information.

Syntax

`auto-virtual-link`

Syntax

This command's *no* form negates the automatic creation of a virtual link:

`no auto-virtual-link`

Mode

OSPF Router configuration: **`XSR(config-router)#`**

Example

```
XSR(config-router)#auto-virtual-link
```

database-overflow

This command dynamically limits the size of OSPF Link-State database overflow, a condition where the XSR is unable to maintain the database in its entirety. Typically, database overflow occurs when a router imports a large number of external, Type 5 LSA routes into OSPF. This command lets you control other LSA types as well: 1-4, 7, and 10.

Usually, this problem can be averted by proper configuration of OSPF routers into stub areas or NSSAs, since AS-external LSAs are omitted from this type of Link-State databases. But, in the event of an unexpected database overflow, there is insufficient time to perform this type of isolation.

Syntax

`database-overflow` `[LSA type][option]`

LSA Type:

`asbr-summary`	AS Border Router Summary LSA (Type 4).
`external`	AS External Area LSA (Type 5).

network	Network LSA (Type 2).
nssa-external	NSSA External LSA (Type 7).
opaque-area	Opaque Area LSA (Type 10).
router	Router LSA (Type 1).
summary	Summary LSA (Type 3).
Option*:*limit	Peak number of LSAs accepted before overflow occurs, ranging from -1 to 2,147,483,647.
exit-overflow interval	Interval before XSR tries to exit overflow. Range: 0 to 86,400 seconds.
warning-level	LSA threshold past which a warning of pending overflow is generated, ranging from 0 to 2,147,483,647.

Defaults

- Limit: -1
- Exit External Interval: 0
- Warning Level: 0

Mode

OSPF Router configuration: **`XSR(config-router)#`**

Examples

The following example configures parameters for Type 5 *external* LSA database overflow:

```
XSR(config)#router ospf 1
XSR(config-router)#database-overflow external limit 1000
XSR (config-router)#database-overflow external exit-overflow-interval 3600
XSR(config-router)#database-overflow external warning-level 900
```

The following example configures parameters for Type 2 *network* LSA database overflow:

```
XSR(config)#router ospf 1
XSR(config-router)#database-overflow network limit 1000
XSR (config-router)#database-overflow external exit-overflow-interval 3600
XSR(config-router)#database-overflow external warning-level 900
```

distance (OSPF)

This command defines an administrative distance (route preference) for the OSPF domain. OSPF distances are ranked higher than *connected* or *static* networks but lower than *RIP* networks.

If several routes to the same destination are offered to the Routing Table Manager (RTM) by different protocols, installation is based on the distance of the protocol with the lowest value. You can set the same distance for different protocols (except for *multiple static routes*) with a tiebreak based on default distances.

Refer to the **`distance`** command on page 176 and **`ip route`** command on page 209 for a comparison with OSPF and static routes.

Syntax

distance ospf {*intra* | *inter* | *ext*} *weight*

intra	OSPF *intra*-area routes.
inter	OSPF *inter*-area routes.
ext	OSPF *external* routes.
weight	Administrative distance used by the routing protocol. Range: 1 to 240.

Syntax of the "no" Form

The *no* command resets the administrative distance to the default value for the particular type of routes. If no type of routes is referenced, the distance for all three types of OSPF routes are reset to the default.

no distance OSPF {*intra* | *inter* | *ext*}

Mode

Router configuration: **XSR(config-router)#**

Default

- Distances between 241 and 255 are reserved for internal use.
- The condition of *intra-area* distance is **less than** *inter-area* distance is **less than** *external* distance is always preserved. If you attempt to configure otherwise, the configuration will fail and you will receive a warning message.
- Default distances *must not be the same* for any two routing protocols.
- For default distances, refer to Table 5-2 below.

Table 5-1 Default Administrative Distances

Route Source	Default Distance
Connected	0
Static	1
BGP external	20
OSPF intra	108
OSPF inter	110
OSPF ext	112
RIP	120
BGP internal	200
Reserved	241-255

Example

This example sets the administrative distance for OSPF external routes to *65*. Note that you can do so only if both intra and inter OSPF distances are less than 65, otherwise you will not be permitted to change the value.

```
XSR(config)#router ospf 1
XSR(config-router)#distance ospf ext 65
```

ip ospf cost

This command sets the cost of sending a packet on a interface. Each router interface that participates in OSPF routing is assigned a default cost. This command overwrites the default.

Syntax

ip ospf cost *cost*

cost	Cost of sending a packet ranging from 1 to 65,535.

Syntax of the "no" Form

no ip ospf cost

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

10

Example

The following example sets cost *20* for interface *FastEthernet 1:*

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip ospf cost 20
```

ip ospf dead-interval

This command sets the interval a router must wait to receive a hello packet from its neighbor before determining that the neighbor is out of service.

Syntax

ip ospf dead-interval *seconds*

seconds	Interval that a router must wait to receive the hello packet. It must be the same on neighboring routers (on a specific subnet), but it can vary between subnets. This value is an unsigned integer ranging from 1 to 65,535 seconds.

Syntax of the "no" Form

The *no* form of this command sets the value to the default:

no ip ospf dead-interval

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

Four times the value of the seconds parameter defined in the **ospf hello-interval** command.

Example

The following example sets the dead interval to *20* for FastEthernet port *2:*

```
XSR(config)#interface fastethernet 2
XSR(config-if<F2>)#ip address 172.16.16.1 255.255.255.0
XSR(config-if<F2>)#ip ospf dead-interval 20
```

ip ospf hello-interval

This command sets the number of seconds a router must wait before sending a hello packet to neighbor routers on the interface.

Syntax

ip ospf hello-interval *seconds*

seconds	The hello interval. It must be the same on neighboring routers (on a specific subnet), but it can vary between subnets, ranging from 1 to 65,535 seconds.

Syntax of the "no" Form

The *no* form of this command sets the value to the default:

```
no ip ospf hello-interval
```

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

- 10 seconds for broadcast and point-to-point networks.

Example

The following example sets the hello interval to *5* for interface *FastEthernet 1:*

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip address 172.16.16.1 255.255.255.0
XSR(config-if<F1>)#ip ospf hello-interval 5
```

ip ospf message-digest-key

This command enables/disables OSPF MD5 authentication on an interface to validate OSPF routing updates between neighboring routers.

Syntax

ip ospf message-digest-key *keyid* ***md5*** *key*

keyid	Key identifier on the interface where MD5 authentication is enabled. Valid values are integers from 1 to 255.
key	Password for MD5 authentication to be used with the *keyid*. Valid values are alphanumeric strings of up to 16 characters.

Syntax of the "no" Form

The *no* form of this command removes the password from this router:

no ip ospf message-digest-key *keyid*

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

OSPF MD5 authentication disabled

Example

The following example enables OSPF *MD5* authentication on interface Serial *1/0*, and sets the key identifier at *20*, and the password as *pass1*.

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#ip address 172.16.77.1 255.255.255.0
XSR(config-if<S1/0>)#ip ospf message-digest-key 20 md5 pass1
XSR(config)#router ospf 1
XSR(config-router)#network 172.16.77.1 0.0.0.0 area 10.0.0.0
XSR(config-router)#area 10.0.0.0 authentication message-digest
```

ip ospf passive

This command suppresses OSPF packets from being sent or received over a specified interface.

Syntax

ip ospf passive

Syntax of the "no" Form

This command's *no* form removes the passive action on the interface:

no ip ospf passive

Mode

Interface configuration: **XSR(config-if<*xx*>#**

Example

The following example imposes OSPF passive on Fast Ethernet interface *1:*

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip ospf passive
```

ip ospf poll-interval

This command sets the OSPF polling interval on Multipoint and Point-to-Point interfaces. The default value allows the adjacency to be established per the default Hello interval.

Syntax

ip ospf poll-interval *<interval>*

interval	Poll period, ranging from 1 to 65,535.

Syntax of the “no” Form

The no form of this command removes the poll interval:

```
no ip ospf poll-interval
```

Mode

Interface configuration: **XSR(config-if<xx>)#**

Example

This example configures the poll interval to *12* times the default hello interval (*10* seconds):

```
XSR(config-if<S1/0:0>)#ip ospf poll-interval 120
```

ip ospf priority

This command sets the OSPF priority value for router interfaces. The priority value is communicated between routers by means of hello messages and this value influences the election of a designated router.

Syntax

ip ospf priority *number*

number	Specifies the router priority, ranging from 0 to 255.

Syntax of the “no” Form

The *no* form of this command sets the value to the default:

```
no ip ospf priority
```

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

1

Example

The following example sets OSPF priority to *20* for FastEthernet port *1:*

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip address 172.16.16.1 255.255.255.0
XSR(config-if<F1>)#ip ospf priority 20
```

ip ospf retransmit-interval

This command sets the interval between retransmissions of link state advertisements for adjacencies that belong to this interface.

Syntax

ip ospf retransmit-interval *seconds*

seconds	Sets the retransmit period, ranging from 1 to 3600 seconds.

Syntax of the “no” Form

The *no* form of this command sets the value to the default:

```
no ip ospf retransmit-interval
```

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

5 seconds

Example

The following example sets the retransmit interval for interface FastEthernet *1* to *20*:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip address 172.16.16.1 255.255.255.0
XSR(config-if<F1>)#ip ospf retransmit-interval 20
```

ip ospf transmit-delay

This command sets the interval required to transmit a link state update packet on this interface.

Syntax

ip ospf transmit-delay *seconds*

seconds	Specifies the transmit delay, ranging from 1 to 3600 seconds.

Syntax of the "no" Form

The *no* form of this command sets the value to the default.

no ip ospf transmit-delay

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

1 second

Example

The following example sets the interval required to transmit a link state update packet on interface FastEthernet 1 at *20* seconds:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip address 172.16.16.1 255.255.255.0
XSR(config-if<F1>)#ip ospf transmit-delay 20
```

network

This command identifies and defines *area IDs* for interfaces OSPF runs on.

Syntax

network *address wildcard-mask* **area** *area-id*

address	IP address of a specific interface or a group of interfaces as a function of the wild-card mask.
wildcard-mask	Inverted mask that begins with 0s and end with 1s. The most specific format is 0.0.0.0, which matches one address. The least specific is 255.255.255.255 matching any address.
area-id	Specifies the area-id that the OSPF address range is linked to. Valid values are decimal values or IP addresses.

Syntax of the "no" Form

The *no* form of this command removes OSPF routing for interfaces identified by the *address* and *wildcard-mask* parameters:

no network *address wildcard-mask* **area** *area-id*

Mode

Router configuration: **XSR(config-router)#**

Defaults

- Disabled
- Costs: LAN - 10, Serial - 64

Example

In this example, three routers are configured to run OSPF. Router *R1* and *R3* are internal routers. R1 is internal to area 1, and R3 internal to area 0. *R2* is an Area Border Router (ABR). Enter the following commands on R1:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip address 131.108.1.1 255.255.255.0
XSR(config)#router ospf 1
XSR(config-router)#network 131.108.1.0 0.0.255.255 area 1
```

On R2 (ABR), enter the following commands:

```
XSR(config)#interface fastethernet 2
XSR(config-if<F2>)#ip address 131.108.1.2 255.255.255.0
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#ip address 131.108.2.3 255.255.255.0
XSR(config)#router ospf 1
XSR(config-router)#network 131.108.1.0 0.0.0.255 area 1
XSR(config-router)#network 131.108.2.0 0.0.0.255 area 0
```

On R3, enter the following commands:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#ip address 131.108.2.4 255.255.255.0
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip address 110.0.0.4 255.0.0.0
XSR(config)#router ospf 1
XSR(config-router)#network 131.108.2.0. 0.0.0.255 area 0
XSR(config-router)#network 110.0.0.0 0.255.255.255 area 0
```

redistribute

This command redistributes static or RIP routes into OSPF.

Syntax

redistribute {**rip** | **bgp** | **static** | **connected**}[**metric** *metric-value*][*metric-type 1 | 2*][*route-map-number*][**tag** *tag-value*]

rip	Imports RIP routes.
bgp	Imports BGP routes.
static	Imports static routes.
connected	Imports connected routes.

metric-value	Cost of a route being redistributed into OSPF, ranging from 0 to 16,777,214.
metric-type	OSPF exterior metric type.
1/2	OSPF external Type 1 or 2 metrics.
route-map-number	Number of the associated route map.

Syntax of the "no" Form

The *no* form of this command cancels the redistribution of routes:

no redistribute *from_protocol* [**metric** *metricvalue*]

Mode

Router configuration: **XSR(config-router)#**

Default

Disabled

Examples

This example redistributes static routes from 5 hops away into RIP:

```
XSR(config-router)#router rip
XSR(config-router)#redistribute static 5
```

The following example redistributes *intra*, *inter* and *external* OSPF routes into RIP:

```
XSR(config-router)#redistribute ospf match internal match external
```

The following example imports all OSPF routes into RIP with the default RIP metric of *1*. It is equivalent to the command entered earlier.

```
XSR(config-router)#redistribute ospf
```

router ospf

This command enables the Open Shortest Path First (OSPF) protocol.

Syntax

router ospf *process-id*

process-id	Process ID number.

Syntax of the "no" Form

The *no* form of this command disables OSPF:

no router ospf process-id

Mode

Global configuration: **XSR(config)#**

Next Mode

Router configuration: **XSR(config-router)#**

Default

OSPF disabled

Example

The following example enables OSPF routing:

```
XSR(config)#router ospf 2
XSR(config-router)#
```

summary address

This command summarizes locally-sourced (Type-5) routes on the XSR which are redistributed from other protocols into OSPF. Type-7 translations are not summarized. Other actions implemented include:

- A summary range becomes active if it includes at least one locally sourced route being redistributed into OSPF. If an active summary range is advertised, then a discard route will be installed for the summary range. Conversely, when it becomes inactive, the discard route is removed.
- Activated summary ranges to be advertised will result in a Type-5 Linke-State Announcement (LSA). If they include a NSSA area, then they will also produce a Type-7 LSA for each NSSA area.
- The type/cost of the summary range is the highest type/cost among all included locally-sourced routes. The forwarding address is 0.
- Summary ranges may overlap. So, for a locally-sourced route, the most specific range becomes active.
- Appendix E processing provides a unique link-state ID for all Type-5 LSAs advertised, be they the result of Type-7 to Type-5 translations, summarization or locally-sourced routes which are not summarized.
- A Type-5 LSA generated by translation may supplant a Type-5 LSA originating from a local source. This will not affect what is being generated into a NSSA because translations are not advertised into NSSA areas.
- If for a given prefix, both a summary and a locally-sourced route exist, the summary will be considered superior even if the summary includes only this locally-sourced route.
- Needless reorigination of Type-5 LSAs will be avoided. For example. importing locally-sourced routes which do not change the type/cost of a summary will not result in reorigination of the summary LSA.
- Type-7 translations are not affected by this command. If an overflow condition occurs then both summary ranges and non-summarized routes will be flushed from the AS.

Syntax

summary-address <*ip-address*><*ip-mask*>[**not-advertise**][**tag** <*tag*>]

Syntax of the “no” Form

The *no* form of this command removes summary addressing on the XSR:

no summary-address *<ip-address><ip-mask>*

ip-address ip-mask	Subnet/mask used for the summary range.
not-advertise	Suppress routes in the summary range.
tag	Value used in the generated Type-5 LSA .

Mode

Router configuration: **XSR(config-router)#**

Example

The following example produce a single Type-5 LSA for all routes redistributed into OSPF covered by the prefix *64.0.0.0/8*:

```
XSR(config-router)#summary-address 64.0.0.0 255.0.0.0
```

timers spf

This command changes timer values to fine-tune the OSPF network.

Syntax

timers spf *spf-delay spf-holdtime*

spf-delay	Delay between the receipt of an update and the SPF execution, ranging from 0 to 4,294,967,295 seconds.
spf-holdtime	Minimum interval, in seconds, between two consecutive OSPF calculations. Range: 0 to 65,535. A value of 0 indicates that two consecutive OSPF calculations are performed immediately after the other.

Syntax of the “no” Form

The *no* form of this command restores the default timer values:

no timers spf

Mode

Router configuration: **XSR(config-router)#**

Defaults

- spf-delay: 5
- spf-holdtime: 10

Example

```
XSR(config)#router ospf 1
XSR(config-router)#network 172.15.0.0 0.0.255.255 area 0.0.0.0
XSR(config-router)#timers spf 7 3
```

OSPF Debug and Show Commands

debug ip ospf dr

This command debugs OSPF designated router events. As with all XSR debug commands, it is set to privilege level 15 by default.

Note: This command does not display in running config because it is a debug function. It must be set manually every time the XSR is rebooted.

Syntax

```
debug ip ospf dr
```

Syntax of the "no" Form

The *no* form of this command returns the debug function to the default:

```
no debug ip ospf dr
```

Mode

EXEC configuration: XSR>

Example

The following example indicates the election of a designated router:

```
OSPF: Elect DR. dr:53.53.53.21 bdr:53.53.53.6 GigabitEthernet 2
```

Parameter Descriptions

Elect DR	OSPF DR Election.
dr:53.53.53.21	Designated router.
bdr:53.53.53.6	Backup Designated router.
GigabitEthernet 2	Interface on which the designated router resides.

debug ip ospf packet

This command debugs received and transmitted OSPF packets. As with all XSR debug commands, it is set to privilege level 15 by default.

Note: This command does not display in *running config* because it is a debug function. It must be set manually every time the XSR is rebooted.

Syntax

```
debug ip ospf packet
```

Syntax of the “no” Form

The *no* form of this command returns the debug function to the default:

```
no debug ip ospf packet
```

Mode

EXEC configuration: **XSR>**

Examples

The following example displays a transmitted Hello packet:

```
OSPF: Tx PKT. Hello v:2  t:1 l:44 rid:1.1.1.4 aid:0.0.0.5 chk:fa94 aut:0000 from
GigabitEthernet 2 to 224.0.0.5
```

The following example displays a received Hello packet that *failed* verification because the area ID does not match:

```
OSPF: Rx PKT. Hello v:2  t:1 l:44 rid:10.0.0.1 aid:0.0.0.3 chk:e9a2 aut:0000 from
GigabitEthernet 2 is NOk
```

The following example displays a received Hello packet that *passed* verification:

```
OSPF: Rx PKT. Hello v:2  t:1 l:48 rid:10.0.0.1 aid:0.0.0.5 chk:8846 aut:0000 from
GigabitEthernet 2 is Ok
```

The following example displays a received database description packet:

```
OSPF: Tx PKT. Database v:2  t:2 l:172 rid:1.1.1.4 aid:0.0.0.5 chk:7204 aut:0000
from GigabitEthernet 2 to 53.53.53.21
```

The following example displays a transmitted link state request packet:

```
OSPF: Tx PKT. LS request v:2  t:3 l:228 rid:1.1.1.4 aid:0.0.0.5 chk:99d5 aut:0000
from GigabitEthernet 2 to 53.53.53.21
```

The following example displays a received link state update packet:

```
OSPF: Rx PKT. LS update v:2  t:4 l:96 rid:10.0.0.1 aid:0.0.0.4 chk:7214 aut:0000
from GigabitEthernet 2.2 is Ok
```

The following example displays a transmitted link state acknowledge packet:

```
OSPF: Tx PKT. LS Ack v:2  t:5 l:44 rid:1.1.1.4 aid:0.0.0.5 chk:b63d aut:0000 from
GigabitEthernet 2 to 53.53.53.21
```

Parameter Descriptions

Tx PKT	OSPF Packet transmitted.
Hello	OSPF Hello Packet.
v:2	OSPF Version.
t:1	OSPF Packet Type.
l:44	OSPF Packet length.
rid:1.1.1.4	OSPF Router ID.
aid:0.0.0.5	OSPF Area ID.
chk:fa94	OSPF Packet Checksum.
aut:0000	Authentication.
from GigabitEthernet 2	Outgoing interface.
to 224.0.0.5	Destination IP address.

Rx PKT	OSPF Packet received.
is Ok	OSPF received packet passed verification.
is NOk	OSPF received packet failed verification (i.e., Area ID does not match).
Database	OSPF Database Description Packet.
LS request	OSPF Link State Request Packet.
LS update	OSPF Link State Update Packet.
LS Ack	OSPF Link State Acknowledge Packet.

debug ip ospf lsas

This command debugs OSPF Link State Advertisements (LSAs). As with all XSR debug commands, it is set to privilege level 15 by default.

Note: This command does not display in *running config* because it is a debug function. It must be set manually every time the XSR is rebooted.

Syntax

```
debug ip ospf lsas
```

Syntax of the "no" Form

The *no* form of this command returns the debug function to the default:

```
no debug ip ospf lsas
```

Mode

EXEC configuration: **XSR>**

Examples

The following example displays an LSA added to the database:

```
OSPF: Add LSA. summary, aid:0.0.0.4 age:0000 opt:02 id:53.53.53.0 rid:1.1.1.4
seq:80000001 chk:4867 l:28
```

The following example displays a received *Type 1* (router) LSA:

```
OSPF: Rx LSA. router, nbr:10.0.0.1 age:002f opt:22 id:10.0.0.1 rid:10.0.0.1
seq:800001aa chk:f671 l:36
```

The following example displays a *queue delayed acknowledgement*:

```
<191>May 21 07:52:39 1.1.1.4 OSPF: Queue Delayed Ack. router, nbr:10.0.0.1
age:002f opt:22 id:10.0.0.1 rid:10.0.0.1 seq:800001aa chk:f671 l:36
```

The following example displays an AS border router *Type 4* summary LSA:

```
OSPF: Rx LSA. asbr-summary, nbr:10.0.0.1 age:03e6 opt:02 id:10.0.0.1 rid:1.1.1.4
seq:80000065 chk:3c9f l:28
```

The following example displays a transmitted external *Type 5* LSA from outgoing interface *GigabitEthernet 2*:

```
OSPF: Tx LSA. external, age:017a opt:20 id:13.0.0.0 rid:10.0.0.1 seq:80000088
chk:807a l:36 from GigabitEthernet 2
```

The following example displays a received *LSA* acknowledgement:

```
OSPF: Rx Ack. external, nbr:10.0.0.1 age:017b opt:20 id:13.0.0.0 rid:10.0.0.1
seq:80000088 chk:807a l:36
```

The following example displays an LSA *Updated/Modified* in the database:

```
OSPF: Upd LSA. summary, aid:00000005 age:0000 opt:02 id:1.1.1.3 rid:1.1.1.4
seq:80000099 chk:4a2d l:28
```

The following example displays a retransmitted LSA:

```
OSPF: RTx LSA. summary, nbr:10.0.0.1 age:0000 opt:02 id:2.2.3.0 rid:1.1.1.4
seq:80000097 chk:1f8f l:28
```

Parameter Descriptions

Add LSA	OSPF Lsa Added to database
summary	OSPF Summary LSA
aid:0.0.0.4	OSPF LSA Area id
age:0000	OSPF LSA Age
opt:02	OSPF LSA Options
id:53.53.53.0	OSPF LSA Identifier
rid:1.1.1.4	OSPF LSA Router Id
seq:80000001	OSPF LSA Sequence Number
chk:4867	OSPF LSA Checksum
l:28	OSPF LSA Length
Rx LSA	OSPF LSA Received
router	OSPF Router LSA
Queue Delayed Ack	OSPF Queued Delayed Acknowledgement
asbr-summary	OSPF AS Border Router Summary LSA
Tx LSA	OSPF LSA Transmitted
Rtx LSA	OSPF LSA retransmitted (from retransmission queue)
external	OSPF External LSA
from GigabitEthernet 2	Outgoing interface
Rx Ack	OSPF Received Link State Acknowledgement
Upd LSA	OSPF LSA Updated/Modified in database

debug ip ospf nbr

This command debugs OSPF neighbor events. As with all XSR debug commands, it is set to privilege level 15 by default.

Note: This command does not display in *running config* because it is a debug function. It must be set manually every time the XSR is rebooted.

Syntax

```
debug ip ospf nbr
```

Syntax of the “no” Form

The *no* form of this command returns the debug function to the default:

```
no debug ip ospf nbr
```

Mode

EXEC configuration: **XSR>**

Examples

The following example displays a Transmit Database Description packet:

```
OSPF: Tx DDP. nbr:10.0.0.1 mtu:05dc opt:42 flg:00 seq:00002400 from
GigabitEthernet 2.1
```

The following example displays a received database description packet from incoming interface *GigabitEthernet 2.1 - I:*

```
OSPF: Rx DDP. nbr:10.0.0.1 mtu:05dc opt:42 flg:03 seq:00002401 from
GigabitEthernet 2.1
```

The following example displays a Neighbor Changing state where the neighbor router ID is *10.0.0.1*, the neighbor IP address is *2.2.3.21*, and the previous state is *EXCHANGE*.

```
OSPF: NBR change state. nbr:10.0.0.1 ipa:1.2.3.21 state:EXCHANGE
```

The following example indicates the neighbor is a slave for the database exchange:

```
OSPF: NBR is slave. nbr:10.0.0.1 ipa:2.2.3.21 state:EX_START
```

Parameter Descriptions

Tx DDP	OSPF Transmit Database Description packet
nbr:10.0.0.1	Neighbor IP address
mtu:05dc	Interface MTU
opt:42	Options
flg:00	Flags
seq:00002400	Sequence number
from GigabitEthernet 2.1	Outgoing interface
Rx DDP	OSPF Received Database Description packet
from GigabitEthernet 2.1	Incoming interface
NBR change state	Neighbor Changing state
nbr:10.0.0.1	Neighbor Router ID
ipa:2.2.3.21	Neighbor IP address
state:EXCHANGE	Previous State
NBR is slave	Neighbor is a slave for a database exchange.

show ip ospf

This command, when any debugging type is enabled, displays output about the following types of OSPF information: designated router events, neighbor events, Link State Advertisements (LSAs), and packets.

Syntax

```
show ip ospf
```

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following is sample output when *all* debugging types are enabled:

```
XSR#show ip ospf

Routing Process "ospf 1 " with ID 1.1.1.4
Supports only single TOS(TOS0) route
It is an area border and autonomous system boundary router
Summary Link update interval is 0 seconds.
External Link update interval is 0 seconds.
Debugging enabled for:
        dr
        lsa
        nbr
        packet
Redistributing External Routes from:
        static
Number of areas in this router is 4
Area BACKBONE (0)
    Number of interfaces in this area is 1
    Area has no authentication
    SPF algorithm executed 2 times
    Area ranges are

Area 0.0.0.5
    Number of interfaces in this area is 2
    Area has no authentication
    SPF algorithm executed 2 times
    Area ranges are
     18.0.0.0 255.0.0.0
```

Parameter Descriptions

Routing Process	OSPF process number and router ID.
Supports	TOS support.

It is	OSPF router designation. Valid values: area border, autonomous system boundary, and internal.
Summary Link update interval	Update interval for summary LSAs generated by this router.
External Link update interval	Update interval for external LSAs generated by this router.
Redistributing External Routes from	Valid redistributed routes: static, RIP, OSPF.
Number of areas in this router	Sum of areas this router belongs to followed by types of areas.
Number of interfaces in this area	Sum of interfaces assigned to this area.
Area authentication	Type of authentication used for this area.
SPF algorithm executed	Number of times the SPF algorithm is run on this router for this area.
Area ranges	Summarized area ranges.

show ip ospf border-routers

This command displays information about OSPF internal route table entries to ABRs and ASBRs.

Syntax

show ip ospf border-routers

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following is sample output:

```
XSR>show ip ospf border-routers

OSPF internal Routing Table

Codes: i - Intra-area route, I - Inter-area route

i 192.168.22.1 [64] via 192.168.11.1, Serial1, ABR, Area 0, SPF 10
i 192.168.22.1 [64] via 192.168.11.1, Serial1, ABR, Area 4, SPF 10
i 192.168.44.1 [64] via 192.168.33.1, Serial2, ABR, Area 0, SPF 10
i 192.168.44.1 [64] via 192.168.33.1, Serial2, ABR, Area 2, SPF 7
i 192.168.44.2 [64] via 192.168.33.1, Serial2, ABR, Area 0, SPF 10
i 192.168.44.2 [64] via 192.168.11.1, Serial1, ABR, Area 0, SPF 10
```

Parameter Descriptions

Router ID	OSPF router ID of the destination border router.
Cost	OSPF cost or metric of reaching a border router identified by the router ID.

Next hop	IP address of an interface on a neighboring router identified by the router ID that can be reached.
Router type	Type of destination border router - ABR or ASBR.
Area	ID of the area through which the route to the destination border router identified by the router ID has been learned.
SPF number	Internal number identifying the SPF calculation that resulted in this coute's installation. This number usually corresponds to the number of SPF calculations on this router for an area through which the route was learned.

show ip ospf database

This command displays the link state (LS) database.

Syntax

show ip ospf database
show ip ospf database router [*link-state-id*]
show ip ospf database network [*link-state-id*]
show ip ospf database summary [*link-state-id*]
show ip ospf database asbr-summary [*link-state-id*]
show ip ospf database nssa-external [*link-state-id*]
show ip ospf database database-external [*link-state-id*]
show ip ospf database database-summary

link-state-id	LS identifier. Valid values are IP addresses.
asbr-summary	Selects asbr-summary (Type 4) link status records. Type 4 LS records are shown in their detail format. ASBR summary records are originated by ABRs.
external	Selects external (Type 5) LS records. Type 5 LS records are shown in detailed format. External records are originated by ASBRs.
network	Selects network (Type 2) LS records, to be shown in detailed format. Network records are originated by designated routers.
router	Selects router (Type 1) LS records to be shown in their detailed format. Router records are originated by all routers.
summary	Selects summary (Type 3) LS records to be shown in original format. Summary records are originated by ABRs.
database-summary	Selects a numerical summary of the contents of the LS database displayed.
nssa-external	Selects nssa-external (Type 7) LS records to be shown in detailed format. Type 7 records are originated by ASBRs.

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following are sample responses:

No Parameter

```
XSR>show ip ospf database

OSPF Router with ID(10.1.2.1)

              Displaying   Net Link     States (Area 0.0.0.0)
LinkID        ADV Router   Age          Seq#         Checksum
10.1.1.1      10.0.0.1     0x1          0x80000001   0x61c610.5.6.1
10.1.2.1      0x0          0x80000001   0x927c

              Displaying   Router Link  States (Area 0.0.0.0)

LinkID        ADV Router   Age          Seq#         Checksum  LinkCount
10.0.0.1      10.0.0.1     0x5          0x80000006   0xcb25    2
10.7.7.1      10.7.7.1  0x1          0x80000003   0x3689    2
10.1.2.1      10.1.2.1  0x0          0x80000009   0xcdaa    4

              Displaying   Summary Net Link States (Area 0.0.0.0)
LinkID        ADV Router   Age       Seq#         Checksum
10.5.5.1      10.1.2.1     0x0       0x80000001   0x927c
```

Router Parameter

```
XSR>show ip ospf database router

OSPF Router with ID (192.168.44.1)

   Router Link States (Area 0.0.0.0)

Routing Bit Set on the LSA
LS age:1292
Options: (No TOS-capability, No DC)
LS Type: Router L inks
Link State ID: 192.168.22.1
LS Seq. Number: 80000007
Checksum: 0x185a
Length:72
Area Border Router
   Number of Links: 4
   Link connected to: a Stub Network
   (Link ID) Network/subnet number: 172.14.0.0.
   (Link Data) Network Mask: 255.255.0.0
   Number of TOS metrics: 0
   TOS 0 Metrics: 10
   Link connected to: another router (point-to-point)
   (Link ID) Neighboring Router ID: 192.168.44.2
   (Link Data) Router Interface address: 192.168.22.1
   Number of TOS metrics: 0
    TOS 0 Metrics: 64
 Link connected to: a Stub Network
   (Link ID) Network/subnet number: 192.168.22.0.
   (Link Data) Network Mask: 255.255.255.0
   Number of TOS metrics: 0
   TOS 0 Metrics: 64
 Link connected to: a Virtual Link
   (Link ID) Neighboring Router ID: 192.168.33.2
```

```
(Link Data) Router Interface address: 0.0.0.0
Number of TOS metrics: 0
TOS 0 Metrics: 64
```

Network Parameter

```
XSR>show ip ospf database network

OSPF Router with ID (192.168.44.2)

Net Link States (Area 0.0.0.0)

Routing Bit Set on this LSA
LS age: 332
Options: (No TOS-capability, DC)
LS Type: Network Links
Link State ID: 172.16.150.1 (address of Designated Router)
Advertising Router: 192.168.44.1
LS Seq. Number: 80000004
Checksum: 0xF627
Length: 32
Network mask: /24
Attached Router: 192.168.44.1
Attached Router: 192.168.44.2
```

Summary Parameter: Response

```
XSR>show ip ospf database summary

OSPF Router with ID (192.168.44.2)
Summary Net Link States (Area 0.0.0.0)

Routing Bit Set on this LSA
LS age: 412
Options: (No TOS-capability, DC)
LS Type: Summary Links (Network)
Link State ID: 172.15.0.0 (summary Network Number)
Advertising Router: 192.168.33.2
LS Seq. number: 80000006
Checksum: 0x6A7B
Length: 28
Network Mask: /16
TOS: 0 Metric: 10
```

ASBR-summary Parameter: Response

```
XSR>show ip ospf database asb-summary

OSPF Router with ID (192.168.44.2)

Summary ASB Link States (Area 1)

LS age: 513
Options: (No TOS-capability, No DC)
LS Type: Summary Links (AS Boundary Router address)
```

```
Link State ID: 172.15.0.0 (summary Network Number)
Advertising Router: 192.168.44.2
LS Seq. number: 80000006
Checksum: 0x5ACD
Length: 28
Network Mask: /0
TOS: 0 Metric: 16777215
```

External Parameter Response

```
XSR>show ip ospf database external

OSPF Router with ID (192.168.44.2)
Type-5 AS External Link States

Routing Bit Set on this LSA
LS age: 98
Options: (No TOS-capability, DC)
LS Type: AS External Link
Link State ID: 172.14.0.0 (External Network Number)
Advertising Router: 192.168.33.2
LS Seq. number: 80000003
Checksum: 0x76E0
Length: 36
Network Mask: /16
Metric Type: 2 (Larger than any link state path)
TOS: 0
Metric: 20
Forward Address: 0.0.0.0
External Route Tag: 0
```

NSSA-External Parameter Response

```
XSR>show ip ospf database nssa-external

OSPF Router with ID (192.168.44.1)

Type-7 AS External Link States (Area 2)

Routing Bit Set on this LSA
LS age: 623
Options: (No TOS-capability, No Type 7/5 translation, DC)
LS Type: AS External Link
Link State ID: 172.14.0.0 (External Network Number)
Advertising Router: 192.168.33.2
LS Seq. number: 80000001
Checksum: 0x5971
Length: 36
Network Mask: /16
Metric Type: 2 (Larger than any link state path)
TOS: 0
Metric: 20
```

```
Forward Address: 192.168.33.2
External Route Tag: 0
```

Database-summary Parameter Response

```
XSR>show ip ospf data database-summary

   OSPF Router with ID (192.168.44.1)
AreaID        Router    Network   S-Net  S-ASBR    Type-7 Subtotal  Delete Manage
0.0.0.0       2         0         2      0         N/A    4         0      0
2             2         0         3      0         4      9         1      1
AS External                                              0         0      0
Total         4         0         5      0         4      13
```

Parameter Descriptions

For No Parameter

Link ID	This field varies as a function of LS record type as follows: • *Router link states* - router ID of the router originating the record. • *Network links states* - interface IP address of designated router to the broadcast network. • *Summary link states* - summary network prefix. • *Asbr-summary link states* - router ID of the ASBR. • *External link states* - external network prefix.
ADV Router	Router ID of the router originating the LS record.
Age	Age of the LS record in seconds.
Seq#	Sequence number assigned by OSPF to each LS record at its time of origination.
Checksum	Field in a LS record used to verify the integrity of the contents upon the receipt by another router.
Link count	Applies only to router LS records. Count is equal to or greater than the sum of active OSPF interfaces on the originating router.

For Router Parameter

Routing bit	Set for LSAs originated by other routers.
LSA age	Age of the LS record in seconds.
LS Type	Meaning of Bit settings in the options field.
LS Type	Router links for a router L record.
Link State ID	Originating router ID for a router LSA.
Advertising Router	Originating router ID.
LS Seq Number	Sequence number assigned by OSPF to this LS record at the time of its origination.
Checksum	Field in a LS record used to verify the integrity of its contents upon the receipt by another router.
Length	Length of the LS record in bytes.

Type of router	Type of OSPF router - internal, ABR, and ASBR.
Number of links	Total individual links inside this LS record.
Link connected to	Assumes different values as a function of the connection offered by a router interface (link). These links can be: point-to-point, to a transit network, to a stub network, and to a virtual link with assigned values from 1 to 4, respectively. Different connection types are referred to as different link types.
(Link ID)	Value corresponds to the link type.
Point-to-point	Router ID of the neighboring router.
Transit network	IP address of designated router interface to the network.
Stub network	IP address of network or subnet.
Virtual link	Router ID of the virtual link neighbor.
(Link Data)	Value corresponds to the link type.
Point-to-point link	Originating router interface address to the network.
Transit network	Originating router interface address to the network.
Stub network	Network mask.
Virtual link	Originating router MIB-II ifIndex value for the unnumbered interface. Virtual links are treated as unnumbered point-to-point links..
Number of TOS metrics	Value is 0 due to lack of TOS support.
Metric	Link (interface) cost.

For Network Parameter

Routing bit	Set for LSAs originated by other routers.
LSA age	Age of the LS record in seconds.
Options	Meaning of Bit settings in the options field.
LS Type	Network links for a network LS record.
Link State ID	IP address of designated router port to the network.
Advertising Router	Originating router ID.
LS Seq. Number	Sequence number assigned by OSPF to this LS record at the time of its origination.
Checksum	Field in a LS record used to verify the integrity of the contents upon the receipt by another router.
Length	Length of the LS record in bytes.
Network mask	Mask for network to which designated router is attached.
Attached router	Router ID for all routers attached to the network that are adjacent to the designated router.

For Summary Parameter Display

Routing bit	Set for LSAs originated by other routers.
LSA age	Age of the LS record in seconds.
Options	Meaning of Bit settings in the options field.

LS Type	Summary links (network) for summary LS record.
Link State ID	IP address of the summarized network.
Advertising Router	Originating router ID.
LS Seq. Number	Sequence number assigned by OSPF to this LS record at the time of its origination.
Checksum	Field in a LS record used to verify the integrity of the contents upon the receipt by another router.
Length	Length of the LS record in bytes.
Network mask	Summary mask for the summarized network.
TOS	0 due to non support of TOS.
Metric	Cost to reach summary network from advertising router (ABR).

For ASB-summary Parameter Display

LSA age	Age of the LS record in seconds.
Options	Meaning of Bit settings in the options field.
LS Type	Summary links (AS Boundary Router) for an asb-summary LS record.
Link State ID	Router ID of the ASBR.
Advertising Router	Originating router ID.
LS Seq. Number	Sequence number assigned by OSPF to this LS record at the time of its origination.
Checksum	Field in a LS record used to verify the integrity of the contents upon the receipt by another router.
Length	Length of the LS record in bytes.
Network mask	Router ID for all routers attached to the network that are adjacent with the designated router. Only for the network parameter.
Attached router	Router ID for all routers attached to the network that are adjacent with the designated router. Only for the network parameter.
TOS	0 due to non support of TOS.
Metric	Cost of reaching the ASBR as advertised by the ASBR.

For External Parameter

Routing bit	Set for LSAs originated by other routers.
LSA age	Age of the LS record in seconds.
Options	Meaning of Bit settings in the options field.
LS Type	AS external link for an external LS record.
Link State ID	IP address of the external network.
Advertising Router	Originating router ID (ASBR between the OSPF and non-OSPF domain).
LS Seq. Number	Sequence number assigned by OSPF to this LS record at the time of its origination.
Checksum	Field in a LS record used to verify the integrity of the contents upon receipt by another router.

Length	Length of the LS record in bytes.
Network mask	Mask of the network.
Metric type	OSPF type 1 or 2 metric.
TOS	0 due to non support of TOS.
Metric	Cost to reach external network from advertising router (ASBR).
Forward address	Address to which packets for the advertised external network must be sent. When it is set to 0.0.0.0, it indicates packets must be sent to the advertising router (ASBR).
External route tag	Tag that can be applied to a route by the protocol from which it originates. This tag can be used for route management, but is often left blank.

For NSSA-external Parameter

Routing bit	Set for LSAs originated by other routers.
LSA age	Age of the LS record in seconds.
Options	Meaning of Bit settings in the options field.
LS Type	AS external link for an nssa-external LS record.
Link State ID	IP address of the external network.
Advertising Router	Originating router ID (ASBR between the OSPF and non-OSPF domain).
LS Seq. Number	Sequence number assigned by OSPF to this LS record at the time of its origination.
Checksum	Field in a LS record used to verify the integrity of the contents upon the receipt by another router.
Length	Length of the LS record in bytes.
Network mask	Mask of the network.
Metric type	OSPF type 1 or 2 metric.
TOS	0 due to non support of TOS.
Metric	Cost to reach external network from advertising router (ASBR).
Forward address	Address to which packets for the advertised external network must be sent. When set to 0.0.0.0, it indicates that packets must be sent to the advertising router (ASBR).
External route tag	Tag that can be applied to a route by the originating protocol. It can be used for route management, but often left blank.

For Database-summary Parameter

Area ID	Area identification.
Area ID	Sum of router LS records in each area.
Network	Sum of network LS records in each area.
S-net	Sum of summary LS records in each area.
S-ASBR	Sum of asb-summary LS records in each area.
Type-7	Sum of nssa-external LS records in each area.

AS external	Sum of external LS records.
Subtotal	Subtotal Sum of LS records per area.
Delete	Sum of LS records waiting for deletion from LS DB.
Maxage	Sum of LS records that have reached maximum age.
Total	Sum of LS records in the LS database on XSR.

show ip ospf interface

This command displays interface OSPF-related information, including network type, priority, cost, hello, interval, dead interval.

Syntax

show ip ospf interface [*type*][*number*]

type	Interface type. Valid interface types are interfaces that exist on this router.
number	Interface number. Valid values correspond to the number of a particular interface type present on this router.

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following are sample responses:

```
XSR>show ip ospf interface
FastEthernet1 is UP
  Internet Address  51.51.51.1  Mask  255.255.255.0
   Internet Address  52.52.52.1  Mask  255.255.255.0 secondary
   Internet Address  53.53.53.1  Mask  255.255.255.0 secondary
Area 0.0.0.2
  Router ID 51.51.51.1,Network Type BROADCAST,Cost: 10
  Transmit Delay is 1 sec,State DR,Priority 1
  Designated Router id 51.51.51.1, Interface addr 51.51.51.1
  No backup designated router on this network
  Timer intervals configured, Hello 10,Dead 40,Wait 40,Retransmit 5
    No Hellos (Passive Interface)
   Neighbor Count is 0, Adjacent neighbor count is 0
```

Parameter Descriptions

Internet address	IP address and mask assigned to this interface.
Area	OSPF area to which this interface is assigned.
Router ID	OSPF router ID. OSPF selects the Router ID from one of the IP addresses configured on this router.
No Hellos (Passive Interface)	OSPF Hellos are not sent or received on this interface.

Network type	OSPF network type. Values can be broadcast, non-broadcast, point-to-point, and point-to-multipoint. Refer to the **ip ospf network** command for more information about network type.
Cost	OSPF interface cost. This value is either the default or assigned by means of the **ip ospf cost** command.
Transmit delay	Number in seconds added to the LSA age field at the time of LSA transmission.
State	*Interface* state - not state between neighbors. Valid values: DR, BDR, Drother, point-to-point, point-to-multipoint, down, backup, loopback.
Priority	Interface priority value. Refer to the **ip ospf priority** command for more information on priority.
Designated Router id	Router ID of the designated router on this subnet if a DR exists.
Interface addr	Address of the designated router's interface to this subnet if a DR exists.
Timer intervals configured	Refers to the **ip ospf hello-interval** and **ip ospf dead-interval** commands for hello and dead interval values. The wait timer represents the period that a router waits before initiating a designated router/ backup router election. The wait timer changes when the dead interval changes. Retransmit timer represents the period between successive transmissions of LSAs until acknowledgement is received.
Neighbor count	Sum of neighbors over the interface.
Adjacent neighbor count	Sum of adjacent (FULL state) neighbors on this port.
secondary	Specified secondary IP address.

show ip ospf neighbor

This command displays the state of communication between this router and its neighbor routers.

Syntax

show ip ospf neighbor [*type number*] [*neighbor-id*] [**detail**]

type	Interface type of the selected interface. Valid interface types are interfaces that exist on this router.
number	Interface number of the selected interface. Valid values correspond to the number of a particular interface type present on this router.
neighbor-id	Router ID of the neighbor router that the selected port is on.
detail	Displays more data about neighbors including the area in which they are neighbors, who the designated router/backup router is on the subnet if applicable, and the decimal equivalent of the E-bit value from the hello packet options field.

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following are sample responses:

```
XSR#show ip ospf neighbor
ID            Pri    State  Dead Intvl    Address    Address
10.7.7.1      1      FULL   40            10.5.6.1   FastEthernet6
10.0.0.1      1      FULL   40            10.1.1.1   FastEthernet3

XSR#show ip ospf neighbor detail
Neighbor 10.7.7.1 interface address 10.5.6.1
   In the area 0.0.0.0 via FastEthernet6
   Neighbor priority is 1, state is FULL.
   Options 1
   Dead interval is 40 sec(s)
   Link state retransmission interval is 5 sec(s)
Neighbor 10.0.0.1, interface address 10.1.1.1
   In the area 0.0.0.0 via FastEthernet3
   Neighbor priority is 1, State is FULL
   Options 1
   Dead interval is 40 sec(s)
   Link state retransmission interval is 5 sec(s)
```

Parameter Description

ID	Router ID of the neighbor.
Pri	Priority of the neighbor over this interface.
State	OSPF communication state *with* followed by the interface status *of* the neighbor.
Dead Intvl	Interval this router will wait without receiving a Hello packet from a neighbor before declaring a neighbor as being down.
Address	IP address of the neighbor over the interface (see next field).
Interface	Interface of this router over which it has neighbors identified by the neighbor ID.
In the area	Area over which this router is a neighbor.
Options	Decimal equivalent of the E-bit from the options field. *0* indicates the area is a stub area, *2* indicates the area is capable of accepting external LSAs (not a stub).

show ip ospf virtual-links

This command displays data about virtual links configured on a router.

Syntax

```
show ip ospf virtual-links
```

Mode

EXEC or Global configuration: **`XSR>`** or **`XSR(config)#`**

Sample Output

The following is sample output:

```
XSR>show ip ospf virtual-links
Virtual Link OSPF_VLI to router 192.168.22.1 is up
  Run as demand circuit.
  DoNotAge LSA not allowed (Number of Dcbitless LSA is 2).
  Transit area 4, via interface Serial1, Cost of using 64
  Transmit Delay is 1 sec, State POINT-TO-POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
  Hello due in 00:00:08
  Adjacency State FULL
Virtual Link OSPF_VLO to router 192.168.44.1 is down
  Run as demand circuit
  DoNotAge LSA not allowed (Number of Dcbitless LSA is 2).
  Transit area 2, Cost of using 65535.
  Transmit delay is 1 sec, State DOWN.,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
```

Parameter Descriptions

Virtual link	Name assigned by OSPF, the ID of the virtual link neighbor and the virtual link status - up or down.
Run as	Type of circuit that OSPF considers the virtual link to be.
DoNotAge LSAs not allowed	LSAs with the DoNotAge bit set in the age field are not permitted in the link state database.
Number of Dcbitless LSA	Sum of LSAs without the Demand Circuit (DC) bit set in the options fields in the link state database of the backbone area.
Transit area	ID of the transit area through which a virtual link is set.
Via interface	Interface of this router to the transit area.
Cost of using	Cost to OSPF of routing through the virtual link.
Transmit delay	Period (in seconds) added to the LSA age field when an LSA is sent from this router through the virtual link. The default (1) can be changed during virtual link configuration.
State	One of the OSPF interface states. The interface state assigned to a virtual link is Point-to-Point. Refer to the description of the **`show ip interface`** command for more information.
Timer intervals configured	Timer intervals for a virtual link can be changed from their default values via optional parameters during virtual link configuration.
Hello due	Interval the router expects to get a Hello packet from its virtual link neighbor. Hello messages may be suppressed along virtual links.
Adjacency	State of adjacency between this router and its virtual link neighbor.

RIP Commands

distance (RIP)

This command defines administrative distances (route preference) in the RIP domain. The RIP default ranks higher than all other routed distances.

If several routes to the same destination are offered to the Routing Table Manager (RTM) by different protocols, installation is based on the distance of the protocol with the lowest value. You can set the same distance for different protocols (except for *multiple static routes*) with a tiebreak based on default distances.

Refer to **`distance ospf`** command on page 147 and **`ip route`** on page 209 for comparison with OSPF and static routes.

Syntax

`distance` *`weight`*

`weight`	The RIP administrative distance, ranging from 1 to 240.

Syntax of the “no” Form

The *no* command resets the administrative distance to the default value:

`no distance` *`weight`*

Defaults

- Distances between 241 and 255 are reserved for internal use.
- Default distances *must not be the same* for any two routing protocols.
- Refer to Table 5-2 below for default distances.

Table 5-2 Default Administrative Distances

Route Source	Default Distance
Connected	0
Static	1
BGP external	20
OSPF intra	108
OSPF internal	110
OSPF external	112
RIP	120
BGP internal	200
Reserved	241-255

Mode

Router configuration: **XSR(config-router)#**

Example

The following example sets the RIP administrative distance to *85*:

```
XSR(config)#router rip
XSR(config-router)#distance 85
```

distribute-list

This RIP command filters networks received in updates/suppresses networks from being advertised in updates.

Syntax

distribute-list access-list-number {*in* | *out*} [*type number*]

access-list number	IP access list number, ranging from 1 to 199. The list defines which networks will be sent and suppressed in routing updates.
in	Applies the access list to incoming routing updates.
out	Applies the access list to outgoing routing updates.
type	Interface type: ATM, BRI, Dialer, Fast/GigabitEthernet, Loopback, Multilink, Serial, or VPN.
number	Interface number on which the access list should be applied. If no interface is set, the ACL will be applied to all updates.

Syntax of the “no” Form

The *no* form of this command removes the filter:

no distribute-list access-list-number {*in* | *out*} [*type number*]

Mode

Router configuration: **XSR(config-router)#**

Default

No filter applied

Example

The following example suppresses network *192.5.34.0* from being advertised in updates on *FastEthernet interface 1:*

```
XSR(config)#access-list 1 deny 192.5.34.0 0.0.0.255
XSR(config)#router rip
XSR(config-router)#distribute-list 1 out fastethernet 1
```

Note: This type of filtering might prove problematic in situations where you want to filter an exact route (for RIP v2). For example, if you want to filter route 10.0.0.0/8, a filter set as **`access-list 1 deny 10.0.0.0 0.255.255.255`** will not suffice, because subnets such as 10.0.0.0/9, 10.0.0.0/10 and so on will also be denied. So, to restrict the filter to 10.0.0.0/8 only, configure an extended access list with the following format: **`access-list 101 deny 10.0.0.0 0.0.0.255 255.0.0.0 0.0.0.0`**

ip rip authentication

This command sets or deletes the single authentication key used for RIP authentication on the interface. Authentication can be used only if a key exists. Deleting an existing key disables the use of authentication for RIP.

Syntax

`ip rip authentication key` *`text`*

`text`	Identifies the key. Valid values are strings of 16 characters or less. Spaces can be used if the complete key is bounded by quotations.

Syntax of the “no” Form

The *no* form of this command deletes the specified key and prevents RIP from using authentication:

`no ip rip authentication key` *`text`*

Mode

Interface configuration: **`XSR(config-if<xx>)#`**

Default

No authentication key

Example

The following example sets the authentication mode as text and the key text as *phone* on *FastEthernet port 1:*

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip rip authentication key phone
XSR(config-if<F1>)#ip rip authentication mode text
```

Note: The command refers to one key only, not a key chain.

RIP Example

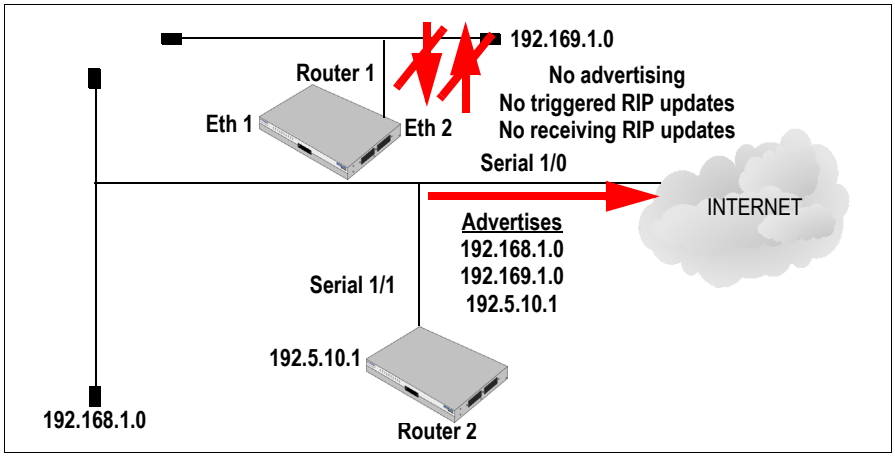
The following example, as shown in Figure 5-2, enables RIP on both FastEthernet interfaces of Router 1, also enabling routing exchanges on the serial link Router 1-Router 2 (Serial port 2).

FastEthernet port 2 is instructed to be totally passive (no advertising on it, no sending of triggered updates, and no receiving of updates).

Serial 1 is allowed to receive both version 1 and 2 RIP, and transmits version 2. The method used is split horizon with poison reverse. Authentication mode text is used on Serial port 1, and the text is *Tex*:

```
XSR(config)#router rip
XSR(config-router)#network 192.168.1.0
XSR(config-router)#network 192.169.1.0
XSR(config-router)#neighbor 192.5.10.1
XSR(config-router)#passive-interface fastethernet 2
XSR(config-router)#no receive-interface fastethernet 2
XSR(config)#interface fastethernet 2
XSR(config-if<F2>)#ip rip disable-triggered-updates
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#ip rip receive version 1 2
XSR(config-if<S1/0>)#ip rip send version 2
XSR(config-if<S1/0>)#ip split-horizon poison
XSR(config-if<S1/0>)#ip rip authentication key Tex
XSR(config-if<S1/0>)#ip rip authentication mode text
```

Figure 5-2 RIP Example



ip rip authentication mode

This command sets the authentication mode used when an authentication key is present.

Syntax

ip rip authentication mode {*text*}

text	Text-only authentication performed.

Syntax of the “no” Form

The *no* form of this command suppresses the use of authentication:

no ip rip authentication mode

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

No authentication mode specified.

Examples

This example sets text authentication mode and the key *XenObhobe* for use on *FastEthernet 1:*

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip rip authentication key XenObhobe
XSR(config-if<F1>)#ip rip authentication mode text
```

The following example enables RIP on both FastEthernet interfaces of router R1, also enabling routing exchanges on the serial link R1-R2 (Serial 2). FastEthernet 2 is instructed to be totally passive (no advertising on it, no sending of triggered updates, and no receiving of updates).

Serial 1/0 is allowed to receive both version 1 and 2 RIP, and transmits version 2. The method used is split horizon with poison reverse. Authentication mode text is used, and the text is *Tex*:

```
XSR(config)#router rip
XSR(config-router)#network 192.168.1.0
XSR(config-router)#network 192.169.1.0
XSR(config-router)#neighbor 192.5.10.1
XSR(config-router)#passive-interface fastethernet 2
XSR(config-router)#no receive-interface fastethernet 2
XSR(config)#interface fastethernet 2
XSR(config-if<F2>)#ip rip disable-triggered-updates
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#ip rip receive version 1 2
XSR(config-if<S1/0>)#ip rip send version 2
XSR(config-if<S1/0>)#ip split-horizon poison
XSR(config-if<S1/0>)#ip rip authentication key Tex
XSR(config-if<S1/0>)#ip rip authentication mode text
```

ip rip disable-triggered-updates

This command prevents RIP from sending triggered updates on the specified interface.

Syntax

```
ip rip disable-triggered-updates
```

Syntax of the “no” Form

```
no ip rip disable-triggered-updates
```

Mode

Interface configuration: **XSR(config-if<xx>)#**

Default

Allows RIP to respond to a triggered update.

Example

This example prevents RIP from responding to a request for triggered updates on F*1*:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip rip disable-triggered-updates
```

ip rip offset

This command adds an offset onto incoming/outgoing metrics to routes learned via RIP.

Syntax

ip rip offset *value*

value	Positive offset to be applied to metrics for networks, ranging from 0 to 16. If the offset is 0, no action is taken.

Syntax of the “no” Form

The *no* form of this command removes an offset:

```
no ip rip offset
```

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

No offset applied

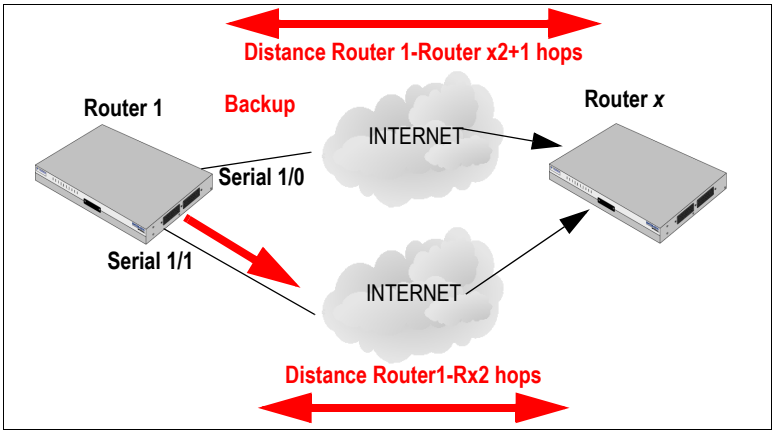
Example

The following example sets an offset of *1* for Serial port *1/0*:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#ip rip offset 1
```

Adding an offset on an interface makes it a backup port. Suppose R1 is only 2 hops away from Rx through both interfaces. By adding 1 to 2 on Serial 1/0, the distance between R1 And Rx through Serial 1/0 becomes 3, making Serial 1/0 a backup.

Figure 5-3 Offset Example



ip rip receive version

This command sets RIP v1 or v2 for update packets received on the port.

Syntax

`ip rip receive version` *`[1] [2]`*

1	RIP version 1.
2	RIP version 2.

Syntax of the “no” Form

The *no* form of this command restores the default version of the RIP module update packets that are accepted on the interface:

```
no ip rip receive version
```

Mode

Interface configuration: **`XSR(config-if<xx>)#`**

Default

Accept both RIP version 1 and 2

Example

This example sets both RIP versions 1 and 2 for update packets received on *FastEthernet port 1:*

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip rip receive version 1 2
```

ip rip send version

This command sets RIP v1 or v2 for update packets sent on the interface.

Syntax

ip rip send version {*1* | *2* | *r1compatible*}

1	RIP version 1.
2	RIP version 2.
r1compatible	Sends version 2 packets, but transmits these as broadcast packets rather than multicast packets, so that systems which only understand RIP version 1 can receive them.

Syntax of the “no” Form

The *no* form restores the version of update packets that was transmitted by the RIP module:

no ip rip send version

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

Version 1

Example

The following example sets RIP version *2* for packets sent on FastEthernet interface *1:*

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip rip send version 2
```

ip split-horizon

This command sets split horizon mode for the packets to be sent by RIP.

Syntax

ip split-horizon

Syntax of the “no” Form

The *no* form of this command disables the split-horizon mechanism entirely:

no ip split-horizon

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

IP split-horizon

Example

The following command sets split horizon for packets to be transmitted by RIP on interface *1*:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip split-horizon
```

neighbor

This command directs the XSR to exchange point-to-point (non-broadcast) routing information with a neighbor. When used in combination with the **passive-interface** command, RIP updates can be exchanged between a subset of routers and access servers on a LAN. One routing update is generated per neighbor.

In the rare case where the XSR or hosts on the LAN segment cannot accept RIP broadcast packets, only configured neighbors will get RIP updates.

Multiple **neighbor** commands can be used to specify additional neighbors or peers.

Syntax

neighbor *neighborAddress*

neighborAddress	IP address of a peer router with which routing datawill be exchanged.

Syntax of the “no” Form

The *no* form of this command disables RIP on the specified interface:

no neighbor *neighborAddress*

Mode

Router configuration: **XSR(config-router)#**

Example

This example instructs the XSR to send RIP updates to all ports on network *192.5.0.0* except interface *F2*. Also, the **neighbor** command allows sending RIP updates specifically to *192.5.10.1.*

```
XSR(config)#router rip
XSR(config-router)#network 192.5.0.0
XSR(config-router)#passive-interface fastethernet 2
XSR(config-router)#neighbor 192.5.10.1
```

network

This command attaches a network of directly connected networks to a RIP routing process.

Syntax

network *netAddress*

netAddress	A directly connected network that RIP will advertise to its neighboring routers. This is an IP address format.

Syntax of the "no" Form

The *no* form of this command disables RIP on the specified interface:

no network *netAddress*

Mode

Router configuration: **XSR(config-router)#**

Example

This example attaches network *192.168.1.0* to the RIP routing process:

```
XSR(config)#router rip
XSR(config-router)#network 192.168.1.0
```

passive-interface

This command prevents RIP from transmitting update packets on an interface (although it can still monitor updates on the interface).

Syntax

passive-interface *type num*

type	Interface types include: ATM, BRI, Dialer, Fast/ GigabitEthernet, Loopback, Multilink, Serial, and VPN.
num	Physical interface number.

Syntax of the "no" Form

The *no* form of this command removes the passive-interface action:

no passive-interface *type num*

Mode

Router configuration: **XSR(config-router)#**

Default

No passive interface

Example

This example sets *F2* as a passive interface. No RIP updates will be transmitted on *F2*:

```
XSR(config-router)#passive-interface fastethernet 2
```

receive-interface

This command allows RIP to receive update packets on an interface. This does not affect the transmission of RIP updates on the specified interface.

Syntax

receive-interface *type num*

type	Interface type.
num	Physical interface number.

Syntax of the "no" Form

no receive-interface *type num*

Mode

Router configuration: **XSR(config-router)#**

Default

Allows the reception of RIP updates on an interface.

Example

The following example *denies* the reception of RIP updates on *F2*:

```
XSR(config-router)#no receive-interface fastethernet 2
```

redistribute (OSPF/Static)

This command redistributes static or OSPF routes into RIP.

Syntax

redistribute {**ospf** | **static**}{**match** *external* [*1* | *2*]| *internal*} **metric** *metricvalue*

ospf	Imports OSPF routes.
static	Imports static routes.
match	Redistributes OSPF routes based on the OSPF type and route metric, ranging from 1 to 16 hops.
external	Redistributes external OSPF routes.
1/2	Redistributes external Type *1* or *2* OSPF routes.
internal	Redistributes *inter-* and *intra*-area OSPF routes.
metric *metricvalue*	Cost of a route being redistributed, ranging from 1 to 16 hops.

Syntax of the “no” Form

The *no* form of this command cancels the redistribution of routes:

no redistribute *from_protocol* [**metric** *metricvalue*]

Mode

Router configuration: **XSR(config-router)#**

Default

Disabled

Examples

This example redistributes static routes from *5* hops away into RIP:

```
XSR(config-router)#router rip
XSR(config-router)#redistribute static 5
```

This example redistributes *intra*, *inter* and *external* OSPF routes into RIP:

```
XSR(config-router)#redistribute ospf match internal match external
```

The following example imports *all OSPF* routes into RIP with the default RIP metric of *1*. It is equivalent to the command entered earlier.

```
XSR(config-router)#redistribute ospf
```

router rip

This command enables/disables the Routing Information Protocol (RIP).

Notes: The XSR supports a total of 750 RIP routing entries with 64 MBytes of memory installed.

RIP commands configured under Interface mode are independent of enabling/disabling the RIP protocol.

Syntax

```
router rip
```

Syntax of the “no” Form

The *no* form of this command disables RIP on the XSR:

```
no router rip
```

Mode

Global configuration: **XSR(config)#**

Next Mode

Router configuration: **XSR(config-router)#**

Example

```
XSR(config)#router rip
XSR(config-router)#
```

timers

This command configures RIP timers.

Syntax

timers basic [*update* | *invalid* | *flush*]

update	Interval the RIP timer is revised, ranging from 1 to 2,147,483,647 seconds.
invalid	Interval the RIP timer is deemed invalid, ranging from 1 to 2,147,483,647 seconds. The invalid interval must be at least three times the update interval.
flush	Interval the RIP timer is flushed, ranging from 1 to 2,147,483,647 seconds. The flush interval must be larger than the invalid interval.

Syntax of the “no” Form

The *no* form of this command resets the timers to the default value:

```
no timers basic
```

Mode

Router configuration: **XSR(config-router)#**

Defaults

- Update: 30 seconds
- Invalid: 180 seconds
- Flush: 300 seconds

Example

The following example sets values for the RIP timers:

```
XSR(config-router)#timers basic 10 30 60
```

RIP Show Commands

show ip rip

This command displays configuration data and statistics global to all ports.

Syntax

show ip rip [*interface* | *database*]

interface	The interface on which RIP is running.
database	The database on which RIP is set up.

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following is a sample response with no option chosen:

```
XSR#show ip rip
Global RIP Stats:

RIP is enabled
RIP timers (in seconds):
       Update interval: 30
       Invalid interval 180
       Flush interval:  300
Routing for Networks:
       172.16.101.1
       172.16.101.5
       172.16.150.0
Route Exchanging Neighbors:
       172.23.11.21
       172.23.11.25
Passive Interfaces:
       FastEthernet 1
Receive Interfaces:
       FastEthernet 1
Distribute List:
       Distribute-list 1 out FastEthernet 1
```

The following is sample output with the *database* option selected:

```
XSR#show ip rip database
   T - triggered on demand
   Directly Connected networks:
       192.168.27.0/24
       192.168.29.0/24
       201.1.1.0/24
       202.1.1.0/24
```

```
   Routing Source Information:
      192.168.28.0/24      via: 192.168.29.22  cost:2    age:16    FastEthernet2
      1.1.1.1/32           via: 192.168.29.22  cost:2    age:16    FastEthernet2
      10.0.0.0/32          via: 201.1.1.0      cost:2    age: -    Serial2/0:1.1
```

The following is sample output with the *interface* option chosen:

```
XSR#show ip rip interface
FastEthernet1 is UP
   Internet Address 10.0.0.0, Mask 255.255.0.0
   Triggered updates are enabled
   Split horizon
   Send rip version is 1
   Receive rip version is 2
   Rip authentication mode is text, key is
   Rip offset metric is 1
Serial1/1 is UP
   Internet Address 11.0.0.0, Mask 255.255.0.0
   Triggered updates are enabled
   Split horizon with poison
   Triggered on demand is enabled
   TRIP number of retransmissions   50
   TRIP polling interval120
   Send rip version is 1
   Receive rip version is 2
   Rip authentication mode is text, key is
   Rip offset metric is 1
```

Parameter Descriptions

Routing for networks	Networks assigned to routing using the **network** command in RIP.
Route Exchanging Neighbors	Neighbors configured to trade routing data used in Pointto-Point exchange of routing data.
Passive Interfaces	Ports RIP will not send update packets on.
Receive Interfaces	Ports RIP will not receive update packets on.
Distribute List	Access list for controlling receive/send updates.
Internet address	IP address and mask assigned to this interface.
Triggered updates	Respond to a request for a trigger update from another router.
Rip versions	Send and receive RIP versions.
Split Horizon	Split horizon mode.
Offset Metric	A value that will be added to routes learned via RIP.

RTP Header Compression Commands

The following commands configures the Real Time Protocol (RTP) header compression on PPP serial interfaces.

The following criteria must be met in order to select packets fro RTP compression

Must be a UDP packet

UDP payload must be less than 500 bytes

Packet must not be fragmented

The destination port of the packet must be within user configured port range (there is no restriction on the source port)

Note: The XSR doesn't impose any restrictions on RTP de-compression.

clear ip rtp header compression interface serial

This command clears the RTP header compression statistics for the specific PPP serial interface.

Syntax

show ip rtp header-compression interface serial *slot/port{.sub-interface}*

slot/port{.sub-interface	The slot, port and sub-interface this command is to be applied to.

Mode

Privileged EXEC: **XSR**

Example

The following example clears the RTP Statistics for serial interface 2/0:1

XSR# clear ip rtp header-compression interface serial 2/0:1

ip rtp compression connections

By default, the software supports a total of 16 RTP header compression connections on the PPP interface. This command will allow the user to change the number of RTP header compression connections in order to specify the total number of RTP header compression connections supported on an interface.

If either end of the PPP link have different max-num-connection values, than the link will negotiate to the lower value.

Syntax

ip rtp compression connections *max-num-connections*

max-num-connections	The max number of RTP connections to be supported on the PPP interface. Range: 3 - 1000

Syntax of the "no" Form

The *no* command resets the RTP header compression connections to the default value of 16:

no rtp compression connections

Default

16 RTP header compression connections on the PPP interface

Mode

Interface configuration: **XSR(config-if<xx>)#**

This command is applicable only on serial interface with PPP encapsulation.

Note: The XSR currently does not block this command on "interface dialer" and on "interface multilink", but the command has no effect on these interfaces.

This command requires a reboot of the interface to take effect.

Example

The following example set the RTP header compression connections to 100, on PPP serial interface S1/0:

```
XSR(config-if<S1/0>)rtp compression connections 100
```

ip rtp header-compression

This command enables or disables the RTP header compression feature on PPP serial interfaces.

The optional passive keyword tells the XSR to compress outgoing RTP packets only if incoming RTP packets on the same interface are compressed.

If you use the command without the passive keyword, the software compresses all RTP traffic.

Note: With this release, XSR now supports both the VJ Header Compression (for TCP and UDP header) and the new IP Header Compression (for TCP, UDP and RTP header compression). XSR cannot be configured to initiates VJ header compression, but it does response to VJ Header compression configuration option from the remote peer with a NAK or REJ.

In this release, the behavior is changed slightly. If RTP is not enabled, then upon receiving a VJ header compression negotiation option, the XSR sends back a NAK or REJ, same as in current release.

Syntax

```
ip rtp header-compression {passive}
```

Parameters

passive	The software compresses outgoing RTP packets only if incoming RTP packets on the same interface are compressed. If the command is used without the passive keyword, the software compresses all RTP traffic.

Syntax of the “no” Form

The *no* command disables the RTP header compression feature:

```
no ip rtp header-compression
```

Default

Disabled

Mode

Interface configuration: **XSR(config-if<xx>)#**

This command is applicable only on serial interface with PPP encapsulation.

Note: The XSR currently does not block this command on "interface dialer" and on "interface multilink", but the command has no effect on these interfaces.

This command requires a reboot of the interface to take effect.

Example

The following example enables RTP header compression on PPP serial interface S1/0:

```
XSR(config-if<S1/0>)#ip rtp header-compression
```

ip rtp range

This command specifies the destination port range of UDP packets used to screen for RTP compression.

Syntax

ip rtp range *starting-port-Num end-Port-Num*

starting-port-Num	Starting destination UDP port number. Range: 1024 to 65535
end-port-Num	Ending Destination UDP port number. Range: 1024 to 65535 Note: The end-port-number must be larger or equal to the starting-port-num.

Syntax of the “no” Form

The *no* command removes the RTP packet ranges

```
no ip rtp range
```

Default

Disabled

Mode

Interface configuration: **XSR(config-if<xx>)#**

This command is applicable only on serial interface with PPP encapsulation.

Note: The XSR currently does not block this command on "interface dialer" and on "interface multilink", but the command has no effect on these interfaces.

Example

The following example set the RTP header range from UDP port 1325 to UDP port 1400, for serial interface S1/0:

```
XSR(config-if<S1/0>)# ip rtp range 325 400
```

show ip rtp header compression interface serial

This command displays the RTP header compression statistics for the specific PPP serial interface.

Note: The existing command "show ppp interface serial" has been updated to add the following line in the PPP stats section "TX/RX IP Header Compression (IPHC is enabled" if IP header compression has been negotiated with the remote peer. See page 8-102 for information on the command "show ppp interface serial".

Syntax

show ip rtp header-compression interface serial *slot/port{.sub-interface}*

slot/port{.sub-interface	The slot, port and sub-interface this command is to be applied to.

Mode

Privileged EXEC: **XSR**

Example

The following example displays the RTP Statistics for serial interface 2/0:1

Router# show ip rtp header-compression interface serial 2/0:1

RTP/UDP/IP Header compression statistics:

Interface Serial 2/0:1

Active/Negotiated connections: RX = 0/0 TX = 0/0

Rcvd:

Compr. RTP = 0 Compr. UDP = 0 Full Header = 0

Error = 0 Dropped = 0 Total Pkts = 0

Bytes rcvd = 0 Bytes Saved = 0 Efficiency Improve = 0.00

Send:

Compr. RTP = 0 Compr. UDP = 0 Full Header = 0

Rej. IP = 0 Rej. Non RTP = 0 Total Pkts = 0

Bytes sent = 0 Bytes Saved = 0 Efficiency Improve = 0.00

Misses = 0 hit Ratio = 0%

Parameter Descriptions

Interface Serial	Type and number of interface.
Active/Negotiated connections:	Number of active and Negotiated RTP connections.

Rcvd:	
Compr. RTP	Number of compressed RTP packets.
Compr. UDP	Number of compressed UDP packets.
Full Header	Number of full header packets received.
Errors	Number of packets that cannot be un-compressed because it is out of sequence, indicating that one or more packets have been lost on the link.
Dropped	Packets whose IP, Port or SSRC does not match that in the received context. These packets are dropped
Total Pkts	Total number of packets received for RTP de-compression
Bytes Rcvd	Total number of bytes received for RTP de-compression
Bytes Saved	Number of bytes saved due to RTP compression.
Efficiency Improve	Efficiency Improvement ratio. Equals (Bytes of actual packet + bytes received) / Bytes Received
Sent	
Compr. RTP	Number of compressed RTP packets.
Compr. UDP	Number of compressed UDP packets. Potential RTP packets with changing x,p and pt fields are sent compressed UDP.
Full Header	Number of full header packets sent.
Rejected IP	Total number of packets that cannot be compressed by RTP compression. These include fragmented packets and packets with IP option fields. These packets are sent uncompressed.
Rejected non RTP	Total number of non RTP packets (RTP version not equal to 2, RTP header length exceeding payload length, SSRC does not match that stored in the TX context. These packets are sent uncompressed.
Total Pkts	Total number of packets sent.
Bytes Rcvd	Total number of bytes sent.
Bytes Saved	Number of bytes saved because of compression.
Efficiency Improve	Efficiency Improvement ratio. Equals (Bytes saved + bytes sent)/ Bytes Sent.
Misses	Number of RTP packets that fails to compress because of no free compression context
Hit ratio	Packets compressed successfully/total packets.

Triggered on Demand RIP Commands

The following commands are subsets of triggered RIP functionality:

- **`ip rip max-retransmissions`** - Specifies the maximum number of retransmissions. Refer to page 190 for the command definition.
- **`ip rip polling-interval`** - Specifies the polling interval for triggered RIP requests. Refer to page 191 for the command definition.

- **ip rip triggered-on-demand** - Enables the functionality on the specified interface. Refer to page 192 for the command definition.

ip rip max-retransmissions

This command sets the maximum number of retransmissions to be sent.

Syntax

ip rip max-retransmissions *number*

number	Number of retransmissions, ranging from 2 to 120.

Syntax of the "no" Form

The *no* command resets the maximum retransmissions value to the default:

no ip rip max-retransmissions

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

36

Example

This example sets the number of retransmissions to 50:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#ip address 1.0.0.0 255.0.0.0
XSR(config-if<S1/0>)#no shutdown
XSR(config-if<S1/0>)#ip rip triggered-on-demand
XSR(config-if<S1/0>)#ip rip max-retransmissions 50
XSR(config)#router rip
XSR(config-router)#network 1.0.0.0
```

ip rip polling-interval

This command sets the polling interval for triggered RIP requests. If a request gets no response after retransmissions peak, requests will continually transmit at intervals set by this command.

Note: The polling interval should be less than the dialer spoofing timeout.

Syntax

ip rip polling-interval *interval*

interval	Polling period ranging from 10 to 600 seconds.

Syntax of the "no" Form

The *no* command resets maximum retransmissions to the default:

no ip rip polling *interval*

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

30 seconds

Example

The following example sets the polling interval to *120* seconds:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#ip address 1.0.0.0 255.0.0.0
XSR(config-if<S1/0>)#no shutdown
XSR(config-if<S1/0>)#ip rip triggered-on-demand
XSR(config-if<S1/0>)#ip rip polling-interval 120
XSR(config)#router rip
XSR(config-router)#network 1.0.0.0
```

ip rip triggered-on-demand

This command enables triggered-on-demand RIP on the specified interface. It is available on a point-to-point Serial (WAN) interface *only.*

On-demand RIP permits the update of an XSR's RIP routing table only when the database changes or when a next hop's reachability is detected on the WAN side of the connection. This functionality reduces the on-demand WAN circuit's routing traffic and allows the link to be brought down when application traffic ceases. Regular RIP updates would prevent the connection from being torn down when application use ends.

On-demand RIP is available under conditions where the route is learned through a *dialer* or *dialer backup* connection and a *dial on demand* link.

The following conditions govern the command's use:

- RIP *must* be enabled.
- IP split horizon *must* be enabled (default). Whether poison is enabled or not, triggered on demand will still send its updates with poison.

Another command, **ip rip disable-triggered-updates**, with the default enforced (triggered updates enabled), invokes triggered updates in a timely fashion as described by RFCs-1058 and 2453 (RIP and RIPv2 protocol) and does not tear down the connection. The two features work independent of each other.

Syntax

ip rip triggered-on-demand

Syntax of the “no” Form

The *no* form of this command disables triggered RIP on the interface:

```
no ip rip triggered-on-demand
```

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

Disabled

Example

The following example configures triggered RIP on Serial port *1/0*:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#ip address 1.0.0.0 255.0.0.0
XSR(config-if<S1/0>)#no shutdown
XSR(config-if<S1/0>)#ip rip triggered-on-demand
XSR(config-router)#network 1.0.0.0
```

Policy-Based Routing Commands

Policy-Based Routing (PBR) on the XSR.

ip policy

This command applies PBR to XSR Fast/GigabitEthernet, Dialer, Loopback, Multilink, VPN and Serial interfaces.

Syntax

```
ip policy
```

Syntax of the “no” Form

The *no* command negates PBR on XSR interfaces:

```
no ip policy
```

Mode

Interface configuration: **XSR(config-if<xx>)#**

Default

Disabled

Examples

The following example enables PBR on interface FastEthernet *2*:

```
XSR(config-if<F2>)#ip policy
```

The following example enables PBR on interface Dialer *57:*

```
XSR(config-if<D57>)#ip policy
```

route-map pbr

This command adds or deletes PBR route-map entries and acquires PBR Map configuration mode. The following commands are subsets of Route Map PBR functionality:

- **match ip address -** Adds/deletes PBR match clauses. See page 5-147 for command definition.
- **set ip next-hop** - Adds or deletes PBR set clauses for the next-hop router. See page 5-147 for command defintion.
- **set interface** - Adds or deletes PBR set clauses on an interface. See page 5-148 for command defintion

Syntax

route-map pbr *sequence-number*

sequence-number	Sequential number of the policy entry in the PBR route map table.

Syntax of the “no” Form

The *no* command deletes the specified policy entry or the whole policy table if no sequence number is specified:

no route-map pbr [*sequence-number*]

Mode

Global configuration: **XSR(config)#**

Next Mode

PBR Map configuration: **XSR(config-pbr-map)#**

Example

In the following example, policy entry number *10* is created:

```
XSR(config)#route-map pbr 10
XSR(config-pbr-map)#
```

match ip address

This command associates the PBR policy with a configured Access Control List (ACL).

Syntax

match ip address *access-number*

access-number	The ACL number used to match traffic.

Syntax of the "no" Form

The *no* command deletes the specified ACL match clause:

no match ip address ***access-number***

Mode

PBR Map configuration: **XSR(config-pbr-map)#**

Example

In the following example, ACL *101* is used to match the traffic:

```
XSR(config-pbr-map)#match ip address 101
```

set ip next-hop

This command specifies a next-hop IP address as the forwarding router for Policy Based Routing.

Syntax

set ip next-hop *ip-address*

ipaddress	IP address of the next hop.

Syntax of the "no" Form

The *no* command deletes the specified set clause:

no set ip next-hope *ip-address*

Mode

PBR Map configuration: **XSR(config-pbr-map)#**

Example

In the following example, *192.168.27.1* is set as the next-hop router:

```
XSR(config-pbr-map)#set ip next-hop 192.168.27.1
```

set interface

This command specifies an XSR interface as the forwarding port for Policy Based Routing.

Syntax

set interface *interface-num*

interface-num	Interface number.

Syntax of the "no" Form

The *no* command deletes the specified set clause:

no set interface *interface-num*

Mode

PBR Map configuration: **XSR(config-pbr-map)#**

Example

The following example sets *F1* as the forwarding interface:

```
XSR(config-pbr-map)#set interface FastEthernet 1
```

PBR Clear and Show Commands

clear ip pbr-cache

This command deletes entries from the PBR cache table.

Syntax

clear ip pbr-cache

Mode

EXEC configuration: **XSR>**

show ip pbr-cache

This command displays the PBR cache that has been built up for fast traffic flow.

Syntax

show ip pbr-cache

Mode

EXEC configuration: **XSR>**

Sample Output

The following is sample output when the command is issued:

```
XSR>show ip pbr-cache
Source        Destination     Age(sec)    IP Prot  TCP/UDP Port  ICMP Code
192.168.1.1   192.168.27.1    109         1         8
192.168.1.1   192.168.27.33   70          255
192.168.1.1   192.168.27.33   50          6         (23, 23)
```

Parameter Descriptions

Source	Source IP address of the packet.
Destination	Destination IP address of the packet.
Age	Seconds left for the lifetime of the cache.
IP Protocol	IP Protocol number.
TCP/UDP Port	TCP/UDP Port number.
ICMP Code	ICMP code number.

show route-map pbr

This command displays the Policy Map Table you have configured. This is the Global Route Map that is used for Policy Based Routing.

Syntax

```
show route-map pbr
```

Mode

EXEC configuration: **XSR>**

Sample Output

The following is sample output when the command is issued:

```
XSR>show route-map pbr
route-map pbr, sequence 10
 Match clauses:
   ip address 102
   ip address 101
 Set clauses:
   next-hop 192.168.27.33
   interface FastEthernet1
```

ARP Commands

arp

This command adds permanent (static) entries to the ARP (Address Resolution Protocol) table. ARP converts an IP address into a physical address. The XSR permits adding/deleting one or all ARP entries.

Syntax

arp *ip-address hardware-address*

ip-address	IP address of a device on the network. Valid values are IP addresses in dotted decimal notation.
hardware-address	The 48-bit hardware address expressed in hexidecimal notation and corresponding to the IP address identified in the *ip-address* parameter.

Syntax of the "no" Form

The *no* form of this command deletes the specified permanent ARP entry:

no arp *ip-address hardware-address*

Mode

Global configuration: **XSR(config)#**

Default

No permanent ARP entries in the ARP table.

Example

The example below adds a permanent ARP entry for the IP address *130.2.3.1:*

```
XSR(config)#arp 130.2.3.1 0003.4712.7a99
```

arp-timeout

This command sets the duration of a dynamic ARP entry in the ARP table before expiring.

Syntax

arp-timeout *seconds*

seconds	Interval that an entry stays in the ARP cache, ranging from 0 to 2,147,483. Zero indicates entries are never cleared from the cache.

Syntax of the "no" Form

The *no* form of his command restores the default value:

no arp-timeout

Mode

Global configuration: **XSR(config)#**

Default

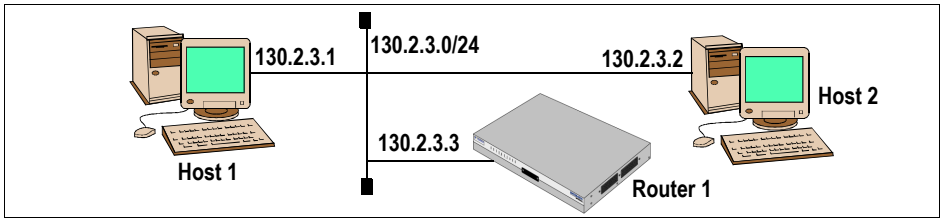
14,400 seconds (4 hours)

Example

This example adds a permanent ARP entry for the IP address *130.2.3.1* and sets the timeout at *5 hours* (18,000 seconds) as shown in Figure 5-4:

```
XSR(config)#arp 130.2.3.1 0003.4712.7a99
XSR(config)#arp-timeout 18000
```

Figure 5-4 ARP Timeout Example



Other IP Commands

ip address

This command sets a primary or secondary IP address on an interface. Secondary IP addresses are allowed on FastEthernet interfaces only. Setting the IP address enables and removing it disables the interface. Before a secondary IP address can be configured, the primary IP address should be configured, and before the primary IP address can be removed, the secondary IP addresses should be removed. This command supports Classless Inter-Domain Routing (CIDR).

Note: When you are routing using the Open Shortest Path First (OSPF) algorithm, be sure that all secondary addresses on an interface fall into the same OSPF area as the primary addresses.

Syntax

ip address {*address mask* | *address&mask* | *negotiated*}{*secondary*]

address	IP address of the interface.
net-mask	Network mask for the configured IP address.
address& mask	Address/mask in format *A.B.C.D./m*, where *A.B.C.D.* is the address, and *m* is the number of bits set to 1 in the mask.
negotiated	IP address negotiated over PPP. BRI, loopback, Fast/ GigabitEthernet and secondary IP interfaces are *not* supported.
secondary	A secondary IP address. If keyword is omitted, the configured address is the primary IP address. *Secondary* is required to add or remove such an address.

Syntax of the “no” Form

The *no* form of this command removes specified IP addresses:

no ip address {*address mask* | *address&mask* | *negotiated*}{*secondary*]

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Examples

The following CIDR example sets IP address *192.168.1.1* with a mask of */24* on interface *F1*.

```
XSR(config)# interface FastEthernet 1
XSR(config-if)# ip address 192.168.1.1/24
```

The following example sets the IP address *192.168.1.1* on *G2*:

```
XSR(config)#interface gigabitethernet 2
XSR(config-if<F1>)#ip address 192.168.1.1 255.255.255.0
```

In the example below, *131.108.1.27* is the primary address and *192.31.7.17* and *192.31.8.17* are secondary addresses for *F1:*

```
XSR(config)#interface FastEthernet 1
XSR(config-if<F1>)#ip address 131.108.1.27 255.255.255.0
XSR(config-if<F1>)#ip add 192.31.7.17 255.255.255.0 secondary
XSR(config-if<F1>)#ip add 192.31.8.17 255.255.255.0 secondary
```

The following example configures *1.1.1.1* as the primary and other IP addresses as secondary addresses for *F1*, removes secondary IP *4.4.4.1* from the interface by entering **no ip address 4.4.4.1 255.255.255.0 secondary**, and updates the primary IP address to *9.9.9.1* by entering **ip address 9.9.9.1 255.255.255.0.**

```
XSR(config)#interface FastEthernet 1
XSR(config-if<F1>)#ip address 1.1.1.1 255.255.255.0
XSR(config-if<F1>)#ip address 2.2.2.1 255.255.255.0 secondary
XSR(config-if<F1>)#ip address 3.3.3.1 255.255.255.0 secondary
XSR(config-if<F1>)#ip address 4.4.4.1 255.255.255.0 secondary
XSR(config-if<F1>)#no shutdown

XSR(config)#interface FastEthernet 1
XSR(config-if<F1>)#no ip address 4.4.4.1 255.255.255.0 secondary

XSR(config)#interface FastEthernet 1
XSR(config-if<F1>)#ip address 9.9.9.1 255.255.255.0
```

ip default-network

This command specifies candidates for the default route and works in conjunction with the **ip route** command which creates static routes to the default network. Default routes must be at least one hop away and have a natural mask attributed to it.

Syntax

ip default-network *network-number*

network-number	Number of the network.

Syntax of the “no” Form

The *no* form of this command removes the route:

no ip default-network *network-number*

Mode

Global configuration: **`XSR(config)#`**

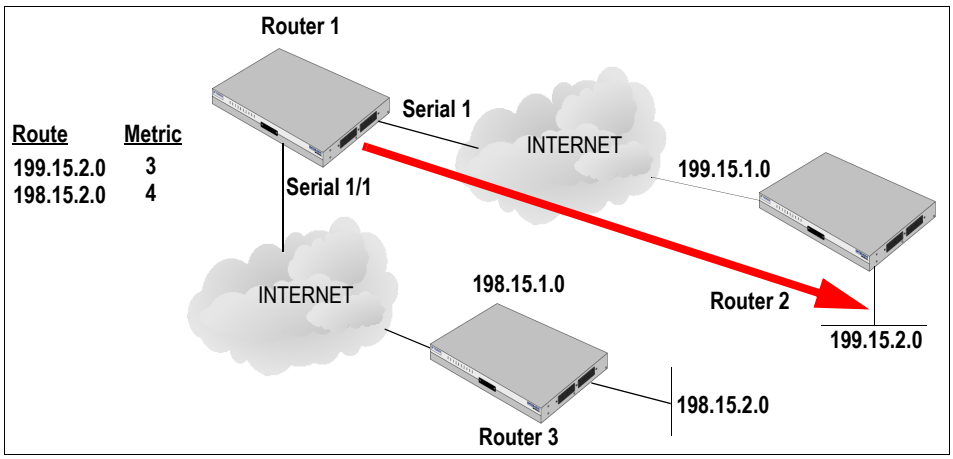
Example

In the following example, as shown in Figure 5-5, Router 1 sets two candidates for the default route: network *199.15.2.0* and *198.15.2.0*.

```
XSR(config)#ip default-network 199.15.2.0
XSR(config)#ip default-network 198.15.2.0
```

Both default routes appear in the routing table, as advertised by *Router 2*, and *Router 3*, which run RIP, so both are candidates for the default route. The route to *199.15.2.0* is three hops away, and the route to *198.15.2.0* is four hops away. So the route to *199.15.2.0* is selected as the default route, and Serial *1/0* is the gateway of last resort for *Router 1*. A default route *0/0* next hop Serial *1/0* is configured on *Router 1*.

Figure 5-5 IP Default Route Example



ip directed-broadcast

This command enables/disables IP directed broadcast. Optionally, you can specify an access list to control which broadcasts are forwarded.

Syntax

`ip directed-broadcast` [*access-list-number*]

Parameters

access-list-number	ACL number. If this is set, a broadcast must pass the ACL to be forwarded. If not set, all broadcasts are forwarded.

Syntax of the “no” Form

The *no* form of this command disables directed broadcast globally:

`no ip directed-broadcast` [*access-list-number*]

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

Enabled

Example

The following example *denies* ICMP broadcasts on port FastEthernet *1:*

```
XSR(config)#access-list 100 deny ICMP any any
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip directed-broadcast 100
```

The following example removes the previous restriction on interface FastEthernet *1* (broadcast will be performed for all protocols):

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#no ip directed-broadcast
```

ip dhcp relay-source gateway

This command allows users to select the source address to use when relaying packets to the DHCP servers. The DHCP servers are configured using **ip helper-address** command.

Syntax

ip dhcp relay-source gateway

Syntax of the “no” Form

The no form negates the command so that the outgoing interface address will be used as the source address:

```
no ip dhcp relay-source gateway
```

Mode

Interface configuration: **XSR(config-if<xx>)#**

Default

The outgoing interface address will be used as the source address.

Example

In the following example, the source address is set for interface fastethernet 1:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip dhcp relay-source gateway
```

ip domain

This command identifies the domain to which the XSR belongs. If the command is reissued, it is considered an update of the domain name and will overwrite the old value with a new value.

The XSR uses the domain name to help create a certificate subject name, which is automatically formated to: *<host name>.<domain name>*. You can configure the host name with the **hostname** command. If the host name is *not set* when you issue the **ip domain** command, the XSR will use the hardcoded *DefaultName*.

Note: For Verisign CA interoperability, you must enter the domain name that you specified when registering with Verisign.

Syntax

ip domain name {*domain-name*}

domain-name	Name of the IP domain to which the XSR belongs. Up to 128 printable characters are permitted with no spaces.

Syntax of the "no" Form

The *no* form of this command resets the IP domain name to no value:

no ip domain name {*domain-name*}

Mode

Global configuration: **XSR(config>#**

Example

In the following example, the domain name *enterasys.com* is used:

```
XSR(config>#ip domain enterasys.com
```

ip equal-cost multi-path

This command enables equal-cost multi-path routing and sets the method for path selection.

Syntax

: for enabling and setting: the selection method:

ip equal-cost multi-path {**round-robin** | **per-flow**}

Parameters

round-robin	Round robin method of selecting the routing path, if multiple paths are available.
per-flow	Per-flow method of selecting the routing path, if multiple paths are available.

Syntax of the "no" Form

The no form of the command disables equal-cost multi-path:

```
no ip equal-cost multi-path
```

Mode

Global configuration: **XSR(config)#**

Default

Disabled

Example

The following example enables equal-cost multi-path and sets the selection method as per-flow:

```
XSR(config)# ip equal-cost multi-path per-flow
```

ip forward-protocol

This command enables broadcast forwarding and specifies which protocols and ports will be forwarded. The IP forward protocol is one of two commands used for UDP broadcast forwarding. Also refer to the **ip helper-address** command, which specifies the new destination.

If a certain service exists inside the node, and there is no need to forward the request to remote networks, the *no* form of this command should be used to disable the forwarding for the specific port. Such requests will not be automatically blocked from being forwarded, just because a service for them exists in the node.

Note: The XSR supports a maximum of 50 IP helper addresses per port and 50 IP forward ports with (64 MBytes of memory installed.

Syntax

ip forward-protocol {*udp* [*port*]}

udp	Forward UDP datagrams.

port	Destination port that controls which UDP services are forwarded. If not set, forwarding is done on the following default ports: • Trivial File Transfer Protocol (TFTP) (port 69) • Domain Naming System (port 53) • Time service (port 37) • NetBIOS Name Server (port 137) • NetBIOS Datagram Server (port 138) • Boot Protocol (BTP) client and server datagrams (ports 67, 68) • TACACS service (port 49) • IEN-116 Name Service (port 42)

Syntax of the "no" Form

The *no* form of this command removes a UDP port or UDP protocol. If the UDP protocol is removed, UDP forwarding is disabled.

no ip forward-protocol {*udp* [*port*]}

Mode

Global configuration: **XSR(config)#**

Defaults

Enabled, but no port specified. This acts as a BOOTP forwarding agent. The above list of ports is used by default for forwarding.

Examples

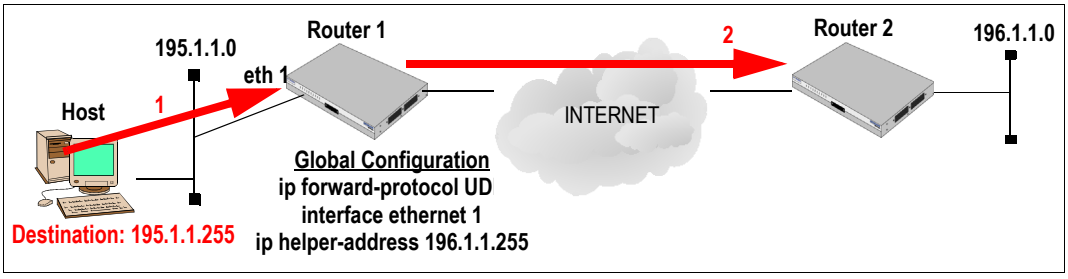
The following example, as shown in Figure 5-6, forwards UDP traffic to a router across the Internet:

```
XSR(config)#ip forward-protocol udp
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip helper-address 196.1.1.255
```

This example removes DNS from the list of ports for which UDP broadcast forwarding is done:

```
XSR(config)#no ip forward-protocol udp 53
```

Figure 5-6 IP Forward-Protocol Example

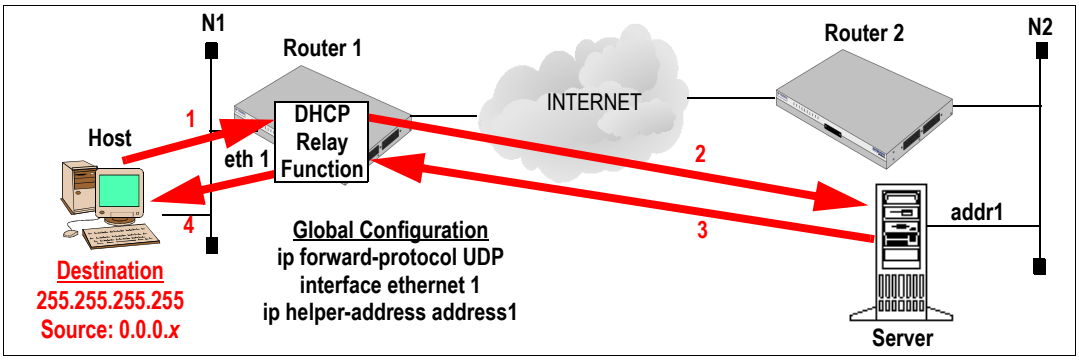


DHCP Relay Functionality

The DHCP Relay functionality is applied with the help of IP broadcast forwarding. A typical situation, as shown in Figure 5-7, occurs when a Host requests an IP address with no DHCP server located on that segment.

Router 1 can forward the DHCP request (1) to the server located on N2, if IP forward-protocol is enabled for UDP, and the address of the DHCP server is configured as a helper address on the receiving interface of Router 1. The DHCP Relay function will detect the DHCP request and make the necessary changes to the header, replacing the destination address with the address of the server, and the source with its own address, and send it further (2) to the server. When the response (3) comes from the server, the DHCP Relay function sends it to the host (4).

Figure 5-7 DHCP Functionality Example



ip helper-address

This command enables forwarding of local broadcasts specifying the new destination address. It is one of two commands used for UDP broadcast forwarding. Also refer to the **`ip forward-protocol`** command which defines the forward protocol and port number. You can add more than one helper address per interface. The command is also used to enable BOOTP Relay.

Syntax

`ip helper-address` *`address`*

`address`	Destination broadcast or host address used when forwarding.

Syntax of the “no” Form

The *no* form disables the forwarding of broadcast packets to the specified address:

`no ip helper-address` *`address`*

Mode

Interface configuration: **`XSR(config-if<xx>)#`**

Example

In this example, with one server on network *191.168.1.255* and the other on network *192.24.1.255*, you permit UDP broadcasts from hosts on either network segment to reach both servers:

```
XSR(config)#ip forward-protocol udp
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip helper-address 192.168.1.255
XSR(config)#interface fastethernet 2
XSR(config-if<F2)#ip helper-address 192.24.1.255
```

ip host

This command defines a static host name-to-address mapping in the static host cache.

Syntax

ip host *name* [*tcp-port-number*] *address*

name	Case-sensitive name of the host.
address	Associated IP address.

Syntax of the "no" Form

Use the *no* form of this command to remove the name-to-address mapping:

no ip host *name address*

Mode

Global configuration: **XSR(config>)#**

Default

Disabled

Example

The following example defines a static mapping for host *ACME*:

```
XSR(config>)#ip host ACME 192.168.57.28
```

ip irdp

This command enables/disables the ICMP Router Discovery Protocol (IRDP), which dynamically discovers routes to other networks, as defined by RFC-1256. IRDP allows hosts to locate routers and can also infer router locations by checking RIP updates. When the XSR operates as a client, router discovery packets are generated.

When the device operates as a host, router discovery packets are received. The IRDP client/server implementation does not actually examine or store full routing tables sent by routing devices, it merely keeps track of which systems are sending such data.

Using IRDP, the XSR can specify both a priority and a period after which a device should be assumed down if no other packets are received.

Syntax

ip irdp [**multicast**|**holdtime** *seconds* | **advertinterval** *seconds* | **preference** *number*]

multicast	:Multicast *address* (224.0.0.1) instead of IP broadcasts.
holdtime *seconds*	The interval router advertisements are held valid, ranging from 1 to 9000 seconds. Value must exceed *advertinterval* but cannot exceed 9000 seconds.
advertinterval *seconds*	Peak interval between router advertisements, ranging from 3 to 1800 seconds.
preference *seconds*	Value from -2147483647 to 2147483647 that sets a router to be the preferred router to which others home. Higher values raise XSR's preference level.

Syntax of the "no" Form

The *no* form of this command disables the IRDP command:

no ip irdp

Defaults

- Multicast: broadcast address
- Holdtime: 1800 seconds
- Advertinterval: 600 seconds
- Preference: 0

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

This example enables IRDP on *F1* with the advertisements and holdtime intervals set to *10* seconds, the preference level set to *10*, and advertisements sent with multicasts:

```
XSR(config-if<F1>)#ip irdp advertinterval 10
XSR(config-if<F1>)#ip irdp holdtime 10
XSR(config-if<F1>)#ip irdp preference 10
XSR(config-if<F1>)#ip irdp multicast
```

ip mtu

This command sets the Maximum Transmit Unit (MTU) size on a port.

Syntax

ip mtu *size*

size	The MTU size, ranging from 68 to 1500 bytes.

Syntax of the "no" Form

The *no* form of this command restores the default value:

no ip mtu

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

1500

Example

The following example sets the MTU size to *1200* for interface Serial *1/0:*

```
XSR(config-if<S1/0>)#ip mtu 1200
```

ip proxy-arp

This command enables/disables Proxy ARP on a per interface basis, allowing the XSR to answer ARP requests on one network for a host on another network. It is available for Fast/GigabitEthernet interfaces only.

Syntax

```
ip proxy-arp
```

Syntax of the “no” Form

The *no* form of this command disables Proxy ARP:

```
no ip proxy-arp
```

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

Enabled

Example

The following example *disables* proxy arp on interface *F1*:

```
XSR(config)#interface fastethernet 1
XSR(config-if)#no ip proxy-arp
```

ip proxy-dns

This command enables Proxy DNS. The XSR’s implementation of this feature supports the configuration of a *forwarding* proxy server which do not perform DNS resolution but pass on and cache DNS queries and replies to other proxy or DNS servers. Use the **show running-config** command to verify current proxy DNS settings on the XSR.

Syntax

ip proxy-dns enable

Syntax of the "no" Form

The *no* form of this command disables Proxy DNS:

no ip proxy-dns enable

Mode

Global configuration: **XSR(config)#**

Default

Disabled

ip proxy-dns name server

This command specifies up to six name servers the proxy DNS server will use.

Syntax

ip proxy-dns name-server *server-address1* [*server-address2...server-address6*]

server-address1	IP address of the name server.
server-address2...server-address6	IP address of additional name servers.

Syntax of the "no" Form

The *no* form of this command removes the configured name server:

no ip proxy-dns name-server *server-address1* [*server-address2...server-address6*]

Mode

Global configuration: **XSR(config)#**

Example

In the following example, *10.10.10.1* is configured as a name server:

```
XSR(config)#ip proxy-dns name-server 10.10.10.1
```

ip redirects

This command enables sending redirect messages if the software is forced to resend a packet through the same interface on which it was received.

Syntax

ip redirects

Syntax of the "no" Form

The *no* form of this command negates IP redirection:

no ip redirects

Default

Enabled

Mode

Global configuration: **XSR(config)#**

Example

In the following example, IP redirection is *disabled*:

XSR(config)#no ip redirects

ip route

This command configures a static IP route.

Note: The XSR supports a maximum of 50 static routes with 64 MBytes of memory installed.

Syntax

ip route {*A.B.C.D. mask*} | {*address&mask*}{address |*interface-type #*}}[*distance*]}

A.B.C.D.	The IP route prefix for the static route destination.
mask	The prefix mask for the static route destination.
address& mask	The forwarding router's IP address and mask, expressed as A.B.C.D./N where A.B.C.D. is the address and N is the number of set bits in the mask..
address	The forwarding router's IP address.
interface-type #	The IP network interface: *ATM, Dialer, Fast/GigabitEthernet, Loopback, Multilink, null,* or *VPN*.
number	Identifies the card and port number: <1-2>/<0-0>, or the card, port and sub-interface number: <1-2>/<0-0>.<1-64>
distance	Administrative metric (preference). Range: ATM (*1* to*255*), BRI *(1* to *240),* Dialer *(0* to *253),* Fast/GigabitEthernet (*1* to *240*), Loopback (*1* to *240*), Multilink *(1-240),* and Serial *(1 to 120).* Only static routes identified by the pair {prefix, mask}, and matching this distance are deleted.

Syntax of the "no" Form

This command's *no* form removes a static route from the routing table:

no ip route {*A.B.C.D. mask*}|{*address&mask*}{address |*interface-type #*}}[*distance*]}

If neither next hop, nor distance is cited, all static routes identified by the pair {prefix, mask} are deleted.

Mode

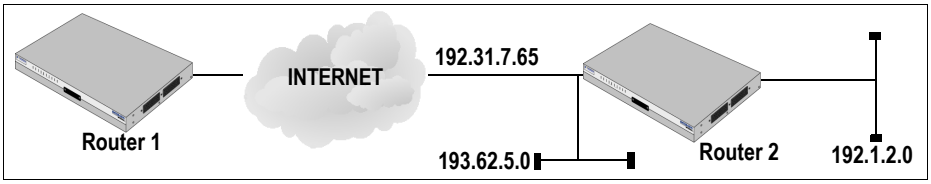
Global configuration: **`XSR(config)#`**

Examples

This example, shown in Figure 5-8, sets 2 static routes to networks 192.1.2.0 and 193.62.5.0 through gateway *192.31.7.65*. Note that the distance is 1 (default), making these routes preferred in case a dynamic routing protocol is running on the same router with its own routes for these destinations.

```
XSR(config)#ip route 192.1.2.0 255.255.255.0 192.31.7.65
XSR(config)#ip route 193.62.5.0 255.255.255.0 192.31.7.65
```

Figure 5-8 Static Route Example



ip route maximum_multiple

This command specifies the maximum number of multiple static routes which are static routes having the same destination but different next hops.

Syntax

`ip route maximum_multiple` *`value`*

`value`	Maximum number of multiple static routes allowed, ranging from 2 to 8.

Syntax of the “no” Form

The *no* form of this command resets the maximum number of multiple static routes to the default:

`no ip route maximum_multiple`

Mode

Global configuration: **`XSR(config)#`**

Default

4

Example

The following example sets the maximum value to *6*:

```
XSR(config)#ip route maximum-multiple 6
```

ip tcp adjust-mss

This command sets the Maximum Segment Size (MSS) for TCP SYN (synchronize) packets. When the XSR terminates PPPoE traffic, a PC connected to the FastEthernet interface may have problems accessing Web sites if the PC's Maximum Transmission Unit (MTU) setting is too high. The MTU contains maximum segment size (MSS) values for TCP packets transmitted by the PC.

Some Web sites do not perform Path MTU discovery correctly. To address this issue, the XSR automatically sets the TCP MSS to 1452 when using PPPoE ports. This forces both TCP peers to send 1492 byte packets so Path MTU discovery never has to deal with PPPoE's 1492-byte MTU.

This is a sub-command of Interface mode and is configured with the following commands:

- **interface fast/gigaethernetx.x**
- **ip address negotiated**
- **encapsulation ppp/mux pppoe**
- **ip mtu 1492**
- **ip tcp adjust-mtu 1400**

Setting the MSS will cause all TCP SYN packets with the MSS option being modified if the option value exceeds the configured MSS.

Syntax

ip tcp adjust-mss *mss*

mss	Range of MSS: 512 to 1452.

Mode

PPPoE Interface configuration: **XSR(config-if)#**

Default

1452 bytes

Example

The following example configures a PPPoE client with an MSS of *1452* bytes on *F1.1:*

```
XSR(config-if<F1.1>)#ip address 192.168.100.1.255.255.255.0
XSR(config-if<F1.1>)#ip tcp adjust-mss 1452
XSR(config-if<F1.1>)#no ip address
XSR(config)#interface dialer 1
XSR(config-if<D1>)#ip address negotiated
XSR(config-if<D1>)#ip mtu 1492
XSR(config-if<D1>)#ip nat outside
XSR(config-if<D1>)#encapsulation ppp
XSR(config-if<D1>)#dialer pool 1
XSR(config-if<D1>)#dialer-group 1
XSR(config-if<D1>)#ppp authentication pap
XSR(config-if<D1>)#ppp pap sent-username frizz password 7 141B1309000528
XSR(config)#ip nat inside source list 101 dialer 1 overload
XSR(config)#ip route 0.0.0.0.0.0.0.0 Dialer1
XSR(config)#access-list 111 permit ip 192.168.100.0.0.0.0.255 any
```

ip telnet server

This command enables or disables Telnet service to the XSR. If the optional parameter is not supplied, the Telnet server is enabled. Since the Telnet server is enabled at boot-up, you must either manually disable it using the CLI or disable it in the *startup-config* file.

Syntax

ip telnet server [*enable* | *disable*]

enable	Enables Telnet service.
disable	Disables Telnet service.

Syntax of the "no" Form

The *no* form of this command disables the Telnet server:

```
no ip telnet server
```

Mode

Global configuration: **XSR(config)#**

Default

Enabled

Example

The following example *disables* the Telnet server:

```
XSR(config)#ip telnet server enable
XSR(config)#no ip telnet server
```

ip unnumbered

This command enables IP processing on a serial interface without assigning an explicit IP address to the interface - it associates a numbered interface whose address will be used with packets originating on this interface. The following conventions are observed:

- If the numbered interface is deleted, the unnumbered association must be deleted as well.
- If the numbered interface changes or deletes its address, the unnumbered association is preserved.
- Routing protocols must be aware of possible changes of the address of the numbered interface they point to, as follows:
  - If the address of the numbered interface is deleted, packets sourced from the unnumbered interface that points to this numbered interface will not be transmitted.
  - If the address of the numbered interface is changed, routing protocols must reevaluate their participation in routing with the unnumbered interfaces. A match between the new address and a configured network must be found for the unnumbered interface to participate in routing.

Syntax

ip unnumbered [*type number*]

type	Type of another interface on which the router has an assigned IP address. It cannot be another unnumbered interface.
number	Number of another interface on which the router has an assigned IP address. It cannot be another unnumbered interface.

Syntax of the "no" Form

The *no* form of this command disables the unnumbered interface:

no ip unnumbered

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

Disabled

Example

In this example, Serial *1* is given *F2*'s address. The serial port is unnumbered:

```
XSR(config-if<F2>)#ip address 145.22.4.67 255.255.255.0
XSR(config)#interface serial 1
XSR(config-if<S1>)#ip unnumbered fastethernet 2
```

ip router-id

This command configures a router identifier, an IPv4 address specified in dotted decimal notation. It is used in routing protocols such as OSPF to uniquely identify a routing instance.

Syntax

ip router-id [*ip-address*]

ip-address	IP Address of router.

Syntax of the "no" Form

The *no* form of this command removes a router identifier:

no ip router-id

Mode

Global configuration: **XSR(config)#**

Example

The following example configures a router identifier:

```
XSR(config)#ip router-id 1.2.3.4
```

IP Clear and Show Commands

clear arp-cache

This command deletes all nonstatic entries from the ARP cache.

Syntax

```
clear arp-cache
```

Mode

Privileged EXEC: **XSR#**

clear ip interface-counters

This command clears all IP interface counters. If you do not enter the optional *type* or *number* value, all interface counters will be erased.

Syntax

clear ip interface-counters [*type*][*number*]

type	Interface type.
number	Interface number.

Mode

Privileged (EXEC): **XSR#**

clear ip proxy-dns cache

This command clears the proxy DNS cache.

Syntax

```
clear ip proxy-dns cache
```

Mode

EXEC: **XSR>**

clear ip traffic-counters

This command clears all IP related counters (IP, ICMP, ARP, UDP, TCP, RIP, OSPF) displayed by the **show ip traffic** command.

Syntax

clear ip traffic-counters

Mode

Privileged EXEC: **XSR#**

clear tcp counters

This command clears all TCP counters.

Syntax

clear tcp counters

Mode

Privileged EXEC: **XSR#**

show ip arp

This command displays all entries in the ARP cache.

Syntax

show ip arp [*ip-address*] [*H.H.H*] [*type number*]

ip-address	ARP entries matching this IP address are displayed.
H.H.H	The 48-bit MAC address.
type number	ARP entries learned via this interface type (Fast/GigabitEthernet) and number are displayed.

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following are sample responses:

```
XSR>show ip arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  134.141.235.251        0   0003.4712.7a99  ARPA   FastEthernet1
Internet  134.141.235.165            0002.1664.a5b3  ARPA   FastEthernet1
Internet  134.141.235.167        4   00d0.cf00.4b74  ARPA   FastEthernet1
```

```
Internet  134.141.235.137         1   00b0.d07f.0cab  ARPA   FastEthernet1
Internet  134.141.235.150         0   00b0.d02c.06d2  ARPA   FastEthernet1
Internet  134.141.235.155         2   00b0.d02c.077e  ARPA   FastEthernet1
Internet  134.141.235.124         17  00b0.d06d.b6ca  ARPA   FastEthernet1
Internet  58.58.58.1              -   0001.f4cc.dd02  ARPA   FastEthernet2
Internet  57.57.57.1              -   0001.f4cc.dd02  ARPA   FastEthernet2
Internet  54.54.54.1              -   0001.f4cc.dd02  ARPA   FastEthernet2
Internet  53.53.53.1              -   0001.f4cc.dd02  ARPA   FastEthernet2
Internet  52.52.52.1              -   0001.f4cc.dd02  ARPA   FastEthernet2
Internet  51.51.51.1              -   0001.f4cc.dd02  ARPA   FastEthernet2

XSR>show ip arp 134.141.235.165
Protocol  Address          Age (min)     Hardware Addr   Type    Interface
Internet  134.141.235.165            -   0002.1664.a5b3  ARPA    FastEthernet1

XSR>show ip arp FastEthernet1
Protocol  Address           Age (min)    Hardware Addr   Type    Interface
Internet  134.141.235.251          0     0003.4712.7a99  ARPA    FastEthernet1
Internet  134.141.235.165          -     0002.1664.a5b3  ARPA    FastEthernet1
Internet  134.141.235.150          2     00b0.d02c.06d2  ARPA    FastEthernet1
Internet  134.141.235.155          5     00b0.d02c.077e  ARPA    FastEthernet1
Internet  134.141.235.124          5     00b0.d06d.b6ca  ARPA    FastEthernet1
```

Parameter Description

Protocol	Type of network address this entry includes.
Address	Network address mapped to the MAC address in this entry.
Age (min)	Interval (in minutes) since this entry was entered in the table, rather than since the entry was last used. The timeout value is 4 hours.
Hardware Addr	MAC address mapped to network address in this entry.
Type	Encapsulation type used for the network address in this entry. Valid values are ARPA (Ethernet encapsulation), SNAP (IEEE 802.3).

show ip interface

Displays the usability status of interfaces configured for IP.

Syntax

show ip interface [*type number*]

type	Interface type: *ATM*, *BRI*, *Dialer*, *Fast/GigabitEthernet*, *Loopback*, *Multilink*, *Serial*, *VPN*, and *Null*. Not specifying a type will display all configured interfaces.
number	Interface number.

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

```
The following is sample output from the command:
XSR>show ip interface
Dialer 0 is Admin Up
  Internet address is 1.1.1.1/24
  Last change: 11:14 AM
  Rcvd:   10245 octets, 1231 unicast packets,
          0 discards, 3 errors, 4 unknown protocol
Sent:     11232 octets, 1132 unicast packets,
          0 discards, 2 errors
  MTU is 1500 bytes
  Proxy ARP is enabled.
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  access list is not set
  Router Discovery is disabled
FastEthernet 0 is Admin Up
  Internet address is 134.141.235.165/24
  Last change: 11:14 AM
  Rcvd:   1245 octets, 131 unicast packets,
          0 discards, 0 errors, 0 unknown protocol
Sent:  11232 octets, 1132 unicast packets,
          0 discards, 2 errors
  MTU is 1500 bytes
  Proxy ARP is enabled.
  Helper address is not set
  Directed broadcast forwarding is enabled
  Outgoing access list is not set
  Inbound  access list is not set
  Router Discovery is enabled
FastEthernet 1 is down
  Internet address is 134.141.234.2/24
  Last change: 11:13 AM
  MTU is 1500 bytes
  Proxy ARP is disabled.
  Helper address is not set
  Directed broadcast forwarding is enabled
  Outgoing access list is not set
  Inbound  access list is not set
 Router Discovery is enabled
```

The following is sample output showing primary and secondary IP addresses:

```
XSR#show ip interface fastEthernet 2
FastEthernet2 is Admin Up
 Internet address is 51.51.51.1, subnet mask is 255.255.255.0
 Internet address is 52.52.52.1, subnet mask is 255.255.255.0 Secondary
 Internet address is 53.53.53.1, subnet mask is 255.255.255.0 Secondary
 Internet address is 54.54.54.1, subnet mask is 255.255.255.0 Secondary
 Internet address is 57.57.57.1, subnet mask is 255.255.255.0 Secondary
```

```
 Internet address is 58.58.58.1, subnet mask is 255.255.255.0 Secondary
 Rcvd: 515027 octets, 3306 unicast packets,
       0 discards, 0 errors, 0 unknown protocol.
 Sent: 363256 octets, 2472 unicast packets,
       0 discards, 0 errors.
 MTU is 1500 bytes.
 Helper address is not set.
 Directed broadcast is enabled.
 Outgoing access list is not set.
 Inbound access list is not set.
 Router discovery is disabled.
```

The following is sample output from a VLAN interface on FastEthernet sub-interface *2.1*:

```
XSR#show ip interface FastEthernet 2.1
FastEthernet2.1 is Admin Up
 Internet address is 1.2.3.4, subnet mask is 255.255.255.0
 Rcvd: 956984 octets, 11 unicast packets,
       0 discards, 0 errors, 0 unknown protocol.
 Sent: 494708 octets, 6789 unicast packets,
       0 discards, 0 errors.
 MTU is 1500 bytes.
 Proxy ARP is enabled.
 Helper address is not set.
 Directed broadcast is enabled.
 Outgoing access list is not set.
 Inbound access list is not set.
 Router discovery is disabled.
 IP Policy Based Routing is not enabled.
```

Parameter Description

FastEthernet 1 is Admin Up	This refers to Layer 3 state for this interface. Valid states are *Up* and *Down*.
Last change	The value of system time when the interface entered the current operational state. If the current state was entered prior to the last re-initialization of the local network management subsystem, then this is 0.
Octets	Sum of octets received/sent through the specified interface.
Unicast packets	Sum of unicast packets received/sent through the port.
Discards	Sum of packets discarded even if no error had been detected, but for internal reasons (for instance to free up some buffer space).
Errors	Sum of packets discarded because of errors.
Unknown protocol	Sum of packets discarded because of unknown or unsupported protocol.
MTU	Shows the MTU value set on the interface.
Proxy ARP	Shows whether proxy ARP is enabled or disabled.
Helper address	Helper address if one has been set.
Directed broadcast forwarding	Indicates whether directed broadcast forwarding is enabled.
Outgoing access list	Indicates whether the interface has an outgoing access list set.

Inbound access list	Indicates whether the interface has an incoming access list set.

show ip irdp

This command displays ICMP router discovery settings.

Syntax

```
show ip irdp
```

Configuration Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following is sample output:

```
XSR>show ip irdp
FastEthernet1 has router server discovery enabled.
Broadcast address is used.
Advertisements will occur between every 450 and 600 seconds.
Advertisements are valid for 1800 seconds.
Preference will be 100.
Serial 1 has router server discovery disabled
FastEthernet2 has router server discovery disabled
```

Parameter Description

Broadcast address is used	Type of addressing used (broadcast or multicast).
Advertisements will occur between every 450 and 600 seconds	Specified minimum and maximum advertising interval for the port.
Advertisements are valid for 1800 seconds	The configured holdtime values for the interface.
Preference is 100	The configured (or in this case default) preference value for the interface.

show ip proxy-dns cache

This command displays the proxy DNS cache.

Syntax

```
show ip proxy-dns cache
```

Mode

EXEC: **XSR>**

Sample Output

The following is sample output from the command:

```
XSR>show ip proxy-dns cache
Name                            Age(sec)
www.enterasys.com               100
www.test.com                    10
```

Parameter Description

Name	Designation of the DNS query.
Age	Seconds remaining for the lifetime of the cache.

show ip route

This command displays information about the Routing Table including route types, IP addresses, and costs. Administrative distances are referenced in each Routing Table entry within the brackets as follows: [*distance/metric*]. The command also displays all alternative routes where more than one route exists to a destination.

Syntax

show ip route [**connected** | *address* [*mask* [**longer-prefixes**]]| **bgp** | **ospf** | **rip** | **static**]

connected	Shows only connected routes.
address	Address about which routing data will be shown.
mask	Argument for a subnet mask.
longer-prefixes	The address and mask pair becomes a prefix and any routes that match the prefix are displayed.
bgp	Shows BGP routes.
ospf	Shows OSPF routes.
rip	Shows RIP routes.
static	Shows static routes.

Note: Bracketed values indicate route distance and cost, where the first value is distance and the second is cost. For example, [120/0003] indicates a distance of 120 (the default distance for RIP) and a cost of 3.

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Defaults

- LAN (FastEthernet 1, 2) interface cost: 10
- Serial interface cost: 64

Sample Output

The following is sample output. Note the route costs as indicated within brackets.

```
XSR>show ip route
Codes: C-connected, S-static, R-RIP, O-OSPF, IA-OSPF interarea
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
* - candidate default, D - default route originated from default net

O E2 222.51.51.0/24     [112/0020] via 192.168.1.6, Dialer1
O IA 192.169.1.0/24     [110/0074] via 192.168.2.9, FastEthernet1
O 192.168.25.0/24       [108/0084] via 192.168.3.9, FastEthernet1
C 192.168.5.0/24        [  0/0001] directly connected, FastEthernet2
R 68.0.0.0/8            [120/0002] via 51.51.51.9, FastEthernet2
R 67.0.0.0/8            [120/0002] via 51.51.51.9, FastEthernet2
R 66.0.0.0/8            [120/0002] via 51.51.51.9, FastEthernet2
C 58.58.58.0/24         [  0/0001] directly connected, FastEthernet2
C 57.57.57.0/24         [  0/0001] directly connected, FastEthernet2
R 55.0.0.0/8            [120/0002] via 51.51.51.9, FastEthernet2
C 54.54.54.0/24         [  0/0001] directly connected, FastEthernet2
C 53.53.53.0/24         [  0/0001] directly connected, FastEthernet2
C 52.52.52.0/24         [  0/0001] directly connected, FastEthernet2
C 51.51.51.0/24         [  0/0001] directly connected, FastEthernet2
S    2.0.0.0/8          [ 65/0001] via 192.168.72.1, FastEthernet1
S    3.0.0.0/8          [  0/0001] directly connected FastEthernet1
```

The following sample output is displayed when IP route *2.0.0.0* is specified:

```
XSR#show ip route 2.0.0.0
    Routing entry for 2.0.0.0 (mask 255.0.0.0)
    Known via "static", distance 65, metric 1
    Redistributing via
    Last update from 192.168.72.1 on FastEthernet1
    Routing Descriptor Blocks:
    *Next hop 192.168.72.1, via FastEthernet1
    Route metric is 1
    Total delay is 0 microseconds, minimum bandwidth is 0kbit
    Reliability  ,  minimum MTU 0 bytes
    Loading , Hops 1
```

Parameter Description

C	Connected route
S	Static route
R	RIP route
O	OSPF route
IA	OSPF interarea route
N1	OSPF NSSA external type 1 route
N2	OSPF NSSA external type 2 route
E1	OSPF external type 1 route

E2	OSPF external type 2 route
*	Candidate default route
D	Default route originated from default network
U	User-configured static route
[x/y]	Distance/metric information
[0060]	Route cost

show ip static database

This command displays static route information including the destination IP *address*, *gateway* IP address, and administrative *distance*.

Syntax

show ip static database [*A.B.C.D. A.B.C.D./mask | interface-type | distance*]

distance	Distance, ranging from 1 to 120 hops.
A.B.C.D.	Next hop
A.B.C.D./<0-32>	IP address and mask
interface-type	XSR interface type: *BRI, Dialer, Loopback, Multilink, Serial , VPN,* or *Fast/GigabitEthernet.*

Mode

EXEC configuration: **XSR>**

Sample Output

The following is sample output:

```
XSR#show ip static database
Maximum number of multiple static routes: 4
Routing Information Sources:
              Address          Gateway       Distance
              7.0.0.0/8        Null0         1
              1.1.1.0/24       2.2.2.2       1
```

Parameter Description

Maximum number of multiple static routes	The maximum number of routes with the same destination but different next hop.
Address	The route.
Gateway	The next hop to reach the address.
Distance	The value of the administrative distance, which is a measure of trustworthiness of the routing update. The lower the value, the more trustworthy the source of the update.

show ip traffic

This command displays general IP protocols statistics.

Syntax

show ip traffic

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following is sample output:

```
XSR>show ip traffic
IP statistics:
 Rcvd:    9040 total, 919 local destination, 7020 to be forwarded
          5 header errors, 45 IP destination not valid
          63 unknown protocol, 0 discards

 Frags:   30 fragments, 10 reassembled, 0 couldn't reassemble
          5 fragmented, 15 fragments, 0 couldn't fragment
 Bcast:   87 received, 97 sent
 Sent:    192 generated,0 drop no route, 0 discards
          0 drop no route, 0 discards

ICMP statistics:
 Rcvd:    44 total
          0 format errors, 0 checksum errors, 0 redirects, 0 unreachable
          2 echo, 2 echo reply, 0 mask requests, 0 mask replies, 0 quench
          0 parameter, 0 timestamp, 0 info replies, 0 time exceeded
 Sent:    23 total
          0 redirects, 23 unreachable, 0 echo, 0 echo reply
          0 mask requests, 0 mask replies, 0 quench, 0 timestamp
          0 info reply, 0 time exceeded, 0 parameter problem

UDP statistics:
 Rcvd:    82858 total, 0 checksum errors, 82852 no port
 Sent:    42 total, 0 forwarded broadcasts

TCP statistics:
 Rcvd:    9138 total, 0 checksum errors, 0 no port
 Sent:    12425 total

RIP statistics:
 Rcvd:    0 total, 0 checksum errors
          0 resp to a query, 0 regular updates, 0 resp triggered by a change
 Sent:    0 total

OSPF statistics:
 Rcvd:    0 total, 0 checksum errors
          0 hello, 0 database desc, 0 link state req
```

```
          0 link state updates, 0 link state acks
 Sent:    0 total

ARP statistics:
 Rcvd:    87441 requests, 5 replies
 Sent:    3 requests, 36 replies (0 proxy)
```

Parameter Description

Total	Sum of datagrams received.
Local destination	Sum of local datagrams successfully delivered to upper layers.
To be forwarded	Sum of input datagrams, for which the XSR is not the destination.
Header errors	Sum of input datagrams discarded due to errors in the IP header, including bad checksum, version number mismatch, ttl exceeded, other format errors.
IP destination not valid	Sum of input datagrams discarded due to IP destination address not valid.
Unknown protocol	Sum of locally addressed datagrams discarded because of unknown or unsupported protocol.
Discards	Sum of input/output datagrams with no problems, but discarded due to internal reasons (such as lack of buffers).
Generated	Sum of packets internally generated.
Drop no route	Sum of packets to be transmitted and dropped because of non existent route to destination.

show resources

This command displays the allowable number of resource entries created and memory utilized. Values displayed reflect the amount of memory installed in your XSR. Monitoring memory usage can help you avoid over-allocating memory to a particular resource and triggering a shortage.

Syntax

```
show resources
```

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following is sample output:

```
XSRtop#show resources

          |Resources|Bytes Per|Total Bytes|Requests
 ( 64MgB)            Resource|InUse     |Resource |InUse       |Denied
                     ========|=========|=========|============|========
       Number of Dynamic ARPs|         1|        96|          96|        0
        Number of Static ARPs|         0|       192|           0|        0
```

Max Unresolved ARP Requests	0	384	0	0
Routing Table Size	3	352	1056	0
Number of Static Routes	2	96	192	0
Number of Secondary IP	0	576	0	0
Number of Virtual IP	0	1344	0	0
IP Helper Addresses	0	96	0	0
UDP Broadcast Fwd Entries	7	96	672	0
OSPF LSA type 1	2	9408	18816	0
OSPF LSA type 2	0	9408	0	0
OSPF LSA type 3	0	320	0	0
OSPF LSA type 4	0	480	0	0
OSPF LSA type 5	0	480	0	0
OSPF LSA type 7	0	576	0	0
Number of ACList Entries	0	192	0	0
Number of Users	1	4000	4000	0
SNMP Read-Only Communities	0	14624	0	0
SNMP Read-Write Communities	1	14816	14816	0
SNMP Trap Servers	0	192	0	0
SNMP users	0	9952	0	0
SNMP groups	2	4672	9344	0
SNMP views	3	3744	11232	0
Number of IP Interfaces	17	7936	134912	0
Number of RIP Net	0	96	0	0
AAA Sessions	0	320	0	0
Authenticated Tunnels	0	640	0	0
IKE/IPsec Tunnels	0	1152	0	0
ISAKMP SA's	0	1920	0	0
IPSEC SA's	0	4448	0	0
L2TP Tunnels	0	5376	0	0
PPTP Tunnels	0	6112	0	0
Dialer Map Classes	1	544	544	0
Dialer Pool size	0	1632	0	0
Frame Relay Map Classes	0	256	0	0
Number of ADSL channels	0	8096	0	0
ISAKMP Proposals	1	96	96	0
Firewall Networks	6	192	1152	0
Firewall Services	0	192	0	0
Firewall Network Groups	0	96	0	0
Firewall Service Groups	0	96	0	0
Firewall Policies	1	320	320	0
Firewall Gating Rules	2	96	192	0
Firewall Filters	2	192	384	0
Firewall Sessions	0	256	0	0
Firewall AuthEntry	0	256	0	0
Crypto Maps	0	736	0	0
PBR Cache Entries	0	96	0	0
Route-map Entries	0	96	0	0
Total:			197824	

Parameter Description

64MgB	Amount of memory installed in the XSR.
Resource	Table, table entry, user, or SNMP category.
ResourcesInUse	Sum of entries currently in use.
Bytes Per Resource	Sum (in bytes) of memory in use by each entry.
Total Bytes InUse	Sum (in bytes) of memory currently used by this resource.

show tcp

This command displays TCP statistics.

Syntax

show tcp {*connections* | *general*}

connections	A summary connections display.
general	A detailed general information display.

Configuration Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following are sample responses:

Connection Table

```
XSR>show tcp connections

-----------TCP Statistics-----------

Current Connections
Local Address          Foreign Address         Connection State
134.141.235.165.23     134.141.235.124.1573    ESTAB
134.141.235.165.23     134.141.235.150.1588    ESTAB
```

General Information Display

```
XSR>show tcp general
-----------TCP Statistics-----------
TCP General Infomation
Maximum number of TCP connections is dynamic
2 connections in state ESTABLISHED or CLOSE-WAIT
Retransmission timeouts: min 220ms; max 684 ms
Rcvd:  870 total
       0 errors
Sent:  701 total
       2 retransmitted
       1 containing the RST flag
0 transitions from CLOSED to SYN-SENT
```

```
4 transitions from LISTEN to SYN-RCVD
2 transitions from SYN-SENT or SYN-RCVD to CLOSED
2 transitions from ESTABLISHED or CLOSE-WAIT to CLOSE
```

Parameter Description

Connection state - Possible states for a TCP connection:

LISTEN	Waiting for a connection request.
SYNSENT	Waiting for a matching connection request after having sent a connection request.
SYNRCVD	Waiting for a confirming connection request ack after having both received and sent a connection request.
ESTAB	Indicates an open connection.
FINWAIT1	Waiting for a connection termination request from the remote TCP host or an ack of the connection termination request previously sent.
FINWAIT2	Waiting for a connection termination request from the remote TCP host.
CLOSEWAIT	Waiting for a connection termination request from local user.
CLOSING	Waiting for a connection termination request ACK from the remote TCP host.
LASTACK	Waiting for an ack of the connection termination request previously sent to the remote TCP host.
TIMEWAIT	Waiting for enough time to pass to be sure the remote TCP host has received the ack of its connection termination request.
CLOSED	Indicates no connection state at all.
Local address	IP address and port of the network server.
Foreign address	IP address and port of the connected remote host .
Retransmission timeout	Retransmission interval of TCP packets that were not acknowledged are waiting for retransmission.

telnet ip_address

This command supports Telnetting to another server.

Syntax

telnet *ip_address* [***port*** *value*]

ip_address	IP address of the server you are Telnetting to.
value	Port number of the Telnet server. Range: from 0 to 65,535.

Mode

Privileged EXEC: **XSR#**

Default

Standard Telnet port 23. If the port is not provided, the client will try to connect to port 23 on the remote server.

Example

The following example connects you to the XSR at *192.57.189.4* via Telnet:

```
XSR#telnet 192.57.189.4 23
```

Network Address Translation Commands

The XSR commands below configure Network Address Translation (NAT).

clear ip nat translation

This command clears dynamic NAT translations from the table before they time out. Although the XSR times out NAT translations by default, it is useful to clear translations before the timeout.

Syntax

clear ip nat translation *interface* {[**all** | *global-ip local-ip*] | [*protocol global-ip global-port local-ip local-port*]}

interface	Port number: Dialer (*0-255*), FastEthernet (*1-2*), Loopback (*0-65535*), Serial (*card/port/channel #*), VPN (*0-255*).
all	Wildcard keyword to clear all dynamic translation entries on an interface.
global-ip	When used without arguments *protocol*, *global-port*, and *local-port*, it clears a simple translation that also contains the specified *local-ip* address. When used with the those arguments it clears an extended translation.
local-ip	Clears an entry that contains this local IP address and the specified *global-ip* address
protocol	Clears an entry containing this protocol and the specified *global-ip* address, *local-ip* address, *global-port* and *local-port*.
global-port	Clears an entry containing this *global-port* and the specified *protocol*, *global-ip* address, *local-ip* address, and *local-port*.
local-port	Clears an entry that contains this local-port and the specified *protocol*, *global-ip* address, *local-ip* address, and *global-port*.

Mode

Privileged EXEC: **XSR#**

Examples

The following example clears are NAT translations on GigabitEthernet interface 2:

XSR#clear ip nat translations g 2 *

2 NAPT entries or NAT mapping removed

The following example clears a specific UDP entry from the NAPT table:

XSR#clear ip nat translation fastEthernet 1 17 200.2.233.1 1220 192.168.27.95 1220

1 NAPT entries or NAT mapping removed

The following example clears all NAPT translations for host 192.168.50.2 on the private network:

XSR#clear ip nat translation fastEthernet 1 192.168.50.2 0.0.0.0

4 NAPT entries or NAT mapping removed

The following example clears all NAPT translations for, to, and from the NATted address of 10.10.10.15:

XSR#clear ip nat translation fastEthernet 1 0.0.0.0 10.10.10.15

5 NAPT entries or NAT mapping removed

ip local pool

This command configures a local pool of IP addresses for distribution to remote peers seeking connection to an interface. The command acquires IP Local Pool mode and makes available this sub-command:

- **exclude** - Bars a range of IP addresses from the local pool. Refer to page 5-184 for the sub-command definition.

Syntax

ip local pool *pool-name IP-address subnet*-mask

pool-name	Name of a particular local address pool.
IP-address	Base address of an IP subnet used to allocate IP addresses.
subnet-mask	Mask of that IP subnet. All subnet address bits matching zero bits in the mask must also be zero; that is, subnet and mask must be zero. May be expressed as **A.B.C.D** or **/<0-32>**.

Note: The pool size (mask) must be */16* or higher (Class B or C) thus limiting any one pool to 64,000 IP addresses.

Syntax of the “no” Form

Use the *no* form of this command to delete an IP address from the pool:

no ip local pool *pool-name*

Mode

Global configuration: **XSR(config)#**

Next Mode

IP Local Pool configuration: **XSR(ip-local-pool)#**

Example

The following example creates local IP address pool *marketing*, which contains all IP addresses in the range *203.57.99.0* to *203.57.99.255*:

```
XSR(config)#ip local pool marketing 203.57.99.0 255.255.255.0
```

exclude

This sub-command bars the use of a range of IP addresses from an earlier created IP pool.

Syntax

exclude {*ip address*}{*number*}

ip address	Starting address to be excluded from pool.
number	Number of addresses to exclude ranging from 1 to 65535.

Syntax of the “no” Form

The *no* form of this command removes the specified IP address from the exclude list:

exclude {*ip address*}{*number*}

Mode

Local IP Pool configuration: **XSR(ip-local-pool)#**

Examples

The following example excludes the ten IP addresses between *192.168.57.100* and *192.168.57.110 f*rom local pool *HQ*:

```
XSR(config)#ip local pool HQ 192.168.57.0 255.255.255.0
XSR(ip-local-pool)#exclude 192.168.57.100 10
```

The following example negates the exclusion of IP addresses *192.168.57.105* and *192.168.57.106* from the earlier excluded range of IP addresses in local pool *HQ*:

```
XSR(config)#ip local pool HQ
XSR(ip-local-pool)#no exclude 192.168.57.105 2
```

ip nat pool

This command defines a pool of IP addresses for Network Address Translation (NAT). NAT pools are configured using the **ip local pool** command and then registered as being used by NAT. A pool must be registered by the XSR or it will not be attached to an interface.

Syntax

ip nat pool *name*

name	Name of the IP local pool.

Syntax of the “no” Form

The *no* command removes one or more addresses from the NAT pool:

no ip nat pool *name*

Mode

Global configuration: **XSR(config)#**

Example

The following example configures the IP NAT pool *NATpool*:

```
XSR(config)#ip nat pool NATpool
```

ip nat service list ???SPTD???

This command specifies a port other than the default port for the File Transfer Protocol (FTP). It is used when you want NAT to pass only FTP control sessions that are using that port. In this case, all client requests using the default port (21) will be dropped by NAT.

Syntax

ip nat service list *access-list-number* **ftp tcp port** *port-number*

list *acl-number*	Standard ACL number, ranging from 1 to 199.
ftp	FTP protocol.
tcp	TCP protocol.
port *port-number*	Port other than the default port. Range: 1 to 65533.

Syntax of the “no” Form

The *no* form of the command disables the port:

no ip nat service list *access-list-number* **ftp tcp port** *port-number*

Mode

Global configuration: **XSR(config)#**

Default

Disabled

Examples

The following example configures non-standard port *2021* for FTP:

```
XSR(config)#ip nat service list 1 ftp tcp port 2021
XSR(config)#access-list 1 permit 10.1.1.1
```

This example sets non-standard port *2021* and standard port *21* for FTP. Be aware that if the FTP server is using both the default and another port, *both* ports must be configured in NAT.

```
XSR(config)#ip nat service list 1 ftp tcp port 21
XSR(config)#ip nat service list 1 ftp tcp port 2021
XSR(config)#access-list 1 permit 10.1.1.1
```

ip nat source (interface mode - NAPT)

This command applies Pool Network Address Translation (NAT) and Network Address Port Translation (NAPT) rules to an XSR interface. Both standard and extended access lists are supported as well as Network Address Port Translation.

Syntax

ip nat source [**list** *access-list-number*]{**assigned overload** | **address** *ip-address* **overload** | **pool** *pool_name* **overload**}

list *access-list-number*	Standard IP ACL number. Packets with source addresses that pass the ACL (permitted by the list) are dynamically translated using the local global address. If the ACL is not specified, the wildcard is assumed.
assigned	IP address of the port used as the source IP address for outgoing packets.
ip-address	Specified arbitrary IP address used as the global NAT IP address.
pool *pool_name*	Group of addresses from which the global address will be chosen.
overload	When overload is specified, the selected global address (either specified or from the pool) will be used to perform NAPT, which ranges from port 20000 to 40960.

Syntax of the “no” Form

The *no* command removes NAT rules from the interface:

no ip nat source [**list** *access-list-number*]{**assigned overload** | **address** *ip-address* **overload** | **pool** *pool_name* **overload**}

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default:

No NAT (rule) specified for the interface.

Example

This example configures Serial interface *1/0* as the source IP address for outgoing packets:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#ip nat source assigned over
```

ip nat source intf-static (interface mode)

This command configures a single static translation entry in the Network Address Translation (NAT) table. Interface static NAT is similar to global NAT; it takes precedence over global static NAT with the implication that if an outgoing/incoming packet matches the interface static NAT no other form of NAT will be performed.

Syntax

ip nat source *[list ACL_number]* **intf-static** {*local-ip global-ip* |{**tcp** | **udp**} *local-ip local-port global-ip global-port*}

list ACL_number	Standard IP ACL number. Packets with source addresses that pass the ACL (permitted by the list) are dynamically translated using the local global address. If the ACL is not specified, the wildcard is assumed.
static	A global static NAT table entry is added.
local-ip	A local IP address assigned to a host on the inside network.
global-ip	Translated IP address.
tcp | **udp**	This value implies that his is a port-specific static NAT.
local-port	Source port of outgoing packets and destination port of incoming packets.
global-port	Destination port of outgoing packets and source port of incoming packets.

Syntax of the “no” Form

The *no* form of the command removes a single static translation entry:

no ip nat intf-source static *local-ip global-ip*

Mode

Interface configuration: **XSR(config-if-<S1>)#**

Example

The following example configures a static NAT system:

```
XSR(config-if<S1>)#ip nat source intf-static 192.178.15.97 10.10.10.5
```

ip nat source static (global mode)

This command configures a single static translation entry in the Network Address Translation (NAT) table. Interface static NAT is similar to global NAT; it takes precedence over global static NAT with the implication that if an outgoing/incoming packet matches the interface static NAT no other form of NAT will be performed.

Syntax

ip nat source static {*local-ip global-ip* |{**tcp** | **udp**} *local-ip local-port global-ip global-port*}

static	A global static NAT table entry is added.
local-ip	A local IP address assigned to a host on the inside network.

global-ip	Translated IP address.
tcp | **udp**	This value implies that his is a port-specific static NAT.
local-port	Source port of outgoing packets and destination port of incoming packets.
global-port	Destination port of outgoing packets and source port of incoming packets.

Syntax of the "no" Form

The *no* form of the command removes a single static translation entry:

no ip nat source static *local-ip global-ip*

Mode

Global configuration: **XSR(config)#**

Example

The following example configures a static NAT system:

```
XSR(config)#ip nat source static 192.178.15.97 10.10.10.5
```

ip nat translation

This command changes the interval after which translations time out.

Syntax

ip nat translation {**timeout** | **udp-timeout** | **tcp-timeout** | **icmp-timeout**}[*seconds*] | [*never*]

timeout	Dynamic NAT interval (not *overload* translations).
udp-timeout	UDP port interval.
tcp-timeout	TCP port interval.
icmp-timeout	ICMP traffic interval.
seconds	Period after which port translation expires.
never	No expiration.

Syntax of the "no" Form

The *no* command configures default timeout values:

no ip nat translation {**timeout** | **udp-timeout** | **tcp-timeout** | **icmp-timeout**} [*seconds*] | [*never*]

Mode

Global configuration: **XSR(config)#**

Defaults

- Timeout: 180 seconds (3 minutes)
- UDP-timeout: 300 seconds (5 minutes)
- TCP-timeout: 86,400 seconds (24 hours)
- ICMP-timeout: 60 seconds

Example

The example below times out UDP port translation entries in *15* minutes:

```
XSR(config)#ip nat translation udp-timeout 900
```

show ip nat translations

This command displays active NAPT translations.

Syntax

show ip nat translations [*interface*]

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following example displays four static NAT entries. Note that external hosts are not tracked for static NAT nor are idle times.

```
XSR#show ip nat translations
Interface GigabitEthernet 2
============================================================
Num Interface-Static NAT : 4
------------------------------------
Pro   Private Host         NAT Addr             External Host     Idle
      (Local IP Addr)      (Global IP Addr)
ANY   146.115.206.31       10.120.112.2         Not Tracked        n/a
TCP   146.115.206.242:80   10.120.112.146:80    Not Tracked        n/a
TCP   146.115.206.242:80   10.120.112.156:80    Not Tracked        n/a
UDP 146.115.206.32:223     10.120.112.156:143   Not Tracked        n/a
```

The following example displays four dynamic NAT entries with assigned address overloading. Note that four different inside hosts appear on the outside with a single NAT IP address (10.10.10.2).

```
XSR#show ip nat translations
```

```
NAPT using address: 10.10.10.2
Num translations: 8
--------------------------------------
Pro   Private Host         NAT Addr             External Host     Idle
      (Local IP Addr)      (Global IP Addr)
UDP   192.168.50.90:1024   10.10.10.2:20002     10.10.10.15:3664   24
UDP   192.168.50.90:1024   10.10.10.2:20001     10.10.10.15:3663   24
UDP   192.168.50.91:1024   10.10.10.2:20004     10.10.10.16:3666   24
UDP   192.168.50.91:1024   10.10.10.2:20003     10.10.10.16:3665   24
TCP   192.168.50.70:1024   10.10.10.2:20006     10.10.15.75:36864  3
TCP   192.168.50.70:1024   10.10.10.2:20005     10.10.15.75:36863  3
TCP   192.168.50.71:1024   10.10.10.2:20008     10.10.15.76:36866  3
TCP   192.168.50.71:1024   10.10.10.2:20007     10.10.15.76:36865  3
```

The following example displays NAT pool entries with overload statistics. Note that a unique NAT IP address is assigned to each internal host and that if there are more internal hosts than the number of addresses in the pool, then multiple internal hosts will share a single NAT address..

```
XSR#show ip nat translations

Pool name: NATPool with overload
ACL Number: 100
--------------------------------------
NAPT using address: 10.10.10.131
Num translations: 2
--------------------------------------
Pro   Private Host           NAT Addr             External Host        Idle
      (Local IP Addr)       (Global IP Addr)
UDP   192.168.50.91:1024    10.10.10.131:20002   10.10.10.16:3666     4
UDP   192.168.50.91:1024    10.10.10.131:20001   10.10.10.16:3665     4
```

Parameter Description

Pro	Protocol of the port identifying the address.
Private Host	The IP address assigned to a host on the inside network.
NAT Addrl	The legitimate IP address.
External Host	Remote host that the packets are destined to.
Idle	Period (seconds) of inactivity of a traffic flow.

Virtual Router Redundancy Protocol Commands

vrrp <group> adver-int

This command configures the interval between successive advertisements sent by the master VR in a virtual group. Advertisements sent by the master VR communicate the state and priority of the current master VR.

Note: All virtual routers in a virtual group must have the same advertisement interval.

Syntax

vrrp *group* **adver-int** [**sec**] *interval*

group	VR group number.
interval	Interval between successive advertisements by master VR. Range: 1- 255 seconds.

Syntax of the "no" Form

Use the *no* form of this command to restore the default value:

no vrrp *group* **adver-int**

Defaults

- Interval: 1 second
- Group: 1 , ranging from 1 to 255

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Examples

The following example sets advertising interval *2* for VR group *2* on FastEthernet interface *1*:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#vrrp 2 adver-int 2
```

The following example sets the default advertising interval for virtual router group *2* on *F1*:

```
XSR(config-if<F1>)#no vrrp 2 adver-int
```

vrrp <group> authentication

This command authenticates Virtual Router Redundancy Protocol (VRRP) packets received from other routers in the group.

When a VRRP packet arrives from another router in the VRRP group, its authentication string inside the packet is compared to the string configured on the local system. If the strings match, the

message is accepted and if not, it is discarded. All routers within the group must share the same authentication string.

Note: Plain text authentication is not meant to be used for security. It simply provides a way to prevent a misconfigured router from participating in the VRRP.

Syntax

vrrp *group* **authentication** *string*

group	Virtual router group number.
string	String (up to 8 alphanumeric characters) to validate incoming VRRP packets.

Syntax of the “no” Form

Disable VRRP authentication by using the *no* form of this command:

no vrrp *group* **authentication**

Defaults

- No authentication of VRRP messages occurs.
- Group : 1, ranging from 1 to 255

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Examples

The following example enables authentication for VR group *1* on *F1*:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#vrrp 1 authentication mypass or vrrp authentication mypass
```

The following example disables authentication:

```
XSR(config-if<F1>)#no vrrp 1 authentication or no vrrp authentication
```

vrrp <group> ip

This command adds up to 11 virtual IP addresses per group and enables a corresponding Virtual Router (VR) on an interface. Be aware of these caveats:

- If the first virtual address for one VR is one of the real addresses in the XSR (it must be on the same port), the next one must also be one of the real addresses (it must be on the same port).
- If the first virtual address is not one of the real addresses on a certain port, the next one must not be one of the real addresses on that port.
- The set of virtual IP addresses configured on each VRRP router belonging to the same group must be the same.

Syntax

vrrp *group* **ip** *ipaddress*

group	VR group number. If you do not specify an input group number, the default group number will be used. Limit: 11 addresses per VR, 44 per router.
ipaddress	IP address of the VR.

Syntax of the "no" Form

The *no* form of this command removes the virtual IP address on a port:

no vrrp *group* ip *ipaddress*

Defaults

- No VR configured
- Group: 1

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Examples

The following example adds and enables virtual *group1* on *F1*. The VRRP group is *1* and IP address *10.0.1.20* is the address of the virtual router.

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#vrrp 1 ip 10.0.1.20 or vrrp 1 ip 10.0.1.20
```

The following example removes virtual IP address *10.0.1.20* from virtual group *1* on *F1*. The VRRP group is *1* and IP address *10.0.1.20* is the address of the virtual router.

```
XSR(config-if<F1>)#no vrrp 1 ip 10.0.1.20 or vrrp ip 10.0.1.20
```

vrrp <group> master-respond-ping

This command allows the Virtual Router (VR) master to respond to an ICMP ping regardless of actual IP address ownership. RFC-2338 specifies that a VR master that is not the actual address owner should not respond to ICMP ping associated with the virtual IP address. This configuration should be consistent on all interfaces participating in a VR.

Syntax

vrrp <*group*> **master-respond-ping**

group	VR group number, ranging from 1 to 255.

Syntax of the "no" Form

The *no* form of this command disables the functionality:

no vrrp *group* **master-respond-ping**

Defaults

- Disabled - the VR master will not respond to an ICMP echo request sent to the virtual IP address if it is not the physical owner.
- If no group is provided, the default group is 1.

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Examples

The following example enables this feature for VR *2* on interface *F1*:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#vrrp 2 master-respond-ping
```

The following example disables this feature for VR *2* on interface *F1*:

```
XSR(config-if<F1>)#no vrrp 2 master-respond-ping
```

vrrp <group> preempt

This command configures the router to take over as master Virtual Router (VR) for a virtual group if it has higher priority than the current master VR.

This feature is enabled by default. You can also configure a delay, which will cause the virtual router to wait the specified interval before issuing an advertisement claiming master ownership.

Notes: The XSR established as the IP address owner will pre-empt another VR, regardless of the setting of this command.

All VRs in a virtual group must share the same preempt attribute. That is, if one VR is set as no preempt, the others must be set likewise.

Syntax

vrrp *group* **preempt** [**delay** <*seconds*>]

group	VR group number.
seconds	Interval the router will delay before issuing an advertisement claiming master ownership.

Syntax of the “no” Form

Disable this feature with the *no* form of the command:

no vrrp *group* preempt

Defaults

- Enabled
- Group : 1 , ranging from 1 to 255
- Seconds: 0

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Examples

The following example enables preempt for virtual router group *1* with a *2*-second delay set on *F1*:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#vrrp 1 preempt delay 2 or vrrp preempt delay 2
```

The following example disables the preempt for VR group *1* on *F1*:

```
XSR(config-if<F1>)#no vrrp 1 preempt or no vrrp preempt
```

vrrp <group> priority

This command sets the priority level of the router within a virtual group. Use it to control which router becomes the master VR.

Syntax

vrrp *group* **priority** *level*

group	VR group number.
level	Priority of the router within the VRRP group. Range: 1 to 254.

Syntax of the “no” Form

The *no* form of this command restores the default value:

no vrrp *group* priority

Defaults

- Level: The priority of the IP address owner is 255, otherwise the default is 100.
- Group: 1 , ranging from 1 to 255

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Examples

This example sets priority *150* for VR group *1* on *F1*:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#vrrp 1 priority 150 or vrrp priority 150
```

The following example sets priority to *default* for VR group *1* on *F1*:

```
XSR(config-if<F1>)#no vrrp 1 priority  or no vrrp priority
```

vrrp <group> track

This command allows a Virtual Router (VR) to track another interface (FastEthernet, Serial, Dialer or Multilink PPP) or one or moe routes on the same router.

When interface A is configured to track interface B, interface A will monitor the status of interface B to decide if it wants to become the master of a VR. When interface B goes down, it will lower its priority to 0 (zero) and refrain from participating in the VR master selection. but will continue to monitor interface B. When interface B comes up, interface A will increase its VR priority back to the original value. If interface A is originally configured as a backup VR, no preemption will occur, but interface A will resume being the backup VR.

This command should be used on the interface that is most likely to be selected as master of the corresponding VR. If the interface is configured as a backup VR, the command has no effect.

When you configure *watchlist* tracking, if all routes fail, the VR will lower its priority to 0 and when at least one of the routes come up, the VR will return to its original priority. When specifying a *watch-group*, be aware that you can use the associated **`dialer watch-list`** command.

Notes: This command should be used on the interface most likely to be chosen master of the corresponding VR. The command has no effect if the interface is configured as a backup VR.

The XSR supports one track interface per VR only, so every time it is configured, the router will overwrite the previous one.

Caution: When you configure the track interface, the VR IP address you specify must be different than the physical IP address of the interface otherwise client ARP tables will not be correctly updated.

Syntax of the “no” Form

`vrrp` *<group>* **`track`** *<interface-type>* **`watch-group`** *watch-list-number*

group	VR group number, ranging from 1 to 255.
interface-type	Name and number of the interface to monitor.
watch-list-number	Number of the Dialer watch-list to monitor, ranging from 1 to 255.

Syntax of the “no” Form

The *no* form of this command disables the functionality:

`no vrrp` *group* **`track`**

Defaults

- No interface tracking.
- If no group is provided, the default group is 1.

Mode

Interface configuration: **`XSR(config-if<xx>)#`**

Example

This example enables the tracking of interface Serial *1/0* by interface *F1* on VR 2:

```
XSR(config)#interface fastethernet 1
XSR(config-if)#vrrp 2 track serial 1/0
```

This example disables the tracking of interface Serial *1/0* by interface *F1* on VR *2*:

```
XSR(config-if)#no vrrp 2 track
```

VRRP Clear and Show Commands

clear vrrp-counters

This command clears statistics for a specified VRRP group; it is governed by the following considerations:

- If you do not specify both group and interface, the statistics for all Virtual Routers (VR) in the VRRP group on this router will be cleared.
- If you specify only the group and not the interface, statistics for all the VRs in the VRRP group whose group ID matches the specified ID on this router will be cleared.
- If you do specify the interface only, statistics for all VRs in the VRRP group configured on this interface on this router will be cleared.
- If you specify both group and interface, only statistics for this specified VRRP group on this router will be cleared.

Syntax

clear vrrp-counters [*group*][*interface*]

group	Virtual router group number, ranging from 1 to 255.
interface	FastEthernet 1 or Fast/GigabitEthernet 2 only.

Mode

EXEC: **XSR>clear vrrp-counters**

Examples

To clear statistics for VR *2* on interface *F1*, enter:

```
XSR#clear vrrp-counters fastethernet 1 2
```

To clear statistics for all the VRs on this router, enter:

```
XSR#clear vrrp-counters
```

show vrrp

This command displays all virtual router information configured on this router.

Syntax

show vrrp

Mode

EXEC: **XSR>**

Sample Output

The following sample output displays configuration data for all virtual routers on this router:

```
XSR#show vrrp

Ethernet Interface:        1
Group ID:                  1
State:                     backup
Preempt:                   Preempt Enabled
Priority:                  100
Adver-int:                 1
Master Down Timer:         3
Authentication Code:       mypass
Virtual IP:                3.3.3.3
Primary IP:                1.1.1.1
Master Router IP:          3.3.3.3
Virtual MAC:               0x00005e005101
BecomeMaster:              2
AdvertiseRcvd:             96
ChecksumErrors:            0
VersionErrrors:            0
PriorityZeroPktsRcvd:      0
PriorityZeroPktsSend:      0
InvalidTypePktsRcvd:       0
UnknownAuthType:           0
AuthTypeErrors:            0
AuthFailures:              0
-------------------------------
Ethernet Interface:        2
Group ID:                  2
State:                     master
Preempt:                   Preempt Enable
Priority:                  100
Adver-int:                 1
Advertise Interval Timer:  1
Authentication Code:       mypass
Virtual IP:                3.3.3.3
Primary IP:                2.2.2.2
Master Router IP:          2.2.2.2
Virtual MAC:               0x00005e005101
BecomeMaster:              2
AdvertiseRcvd:             96
ChecksumErrors:            0
VersionErrrors:            0
PriorityZeroPktsRcvd:      0
PriorityZeroPktsSend:      0
```

```
InvalidTypePktsRcvd:       0
UnknownAuthType:           0
AuthTypeErrors:            10
AuthFailures:              0
```

show vrrp interface

This command displays all the virtual routers and their status on a specified interface.

Syntax

show vrrp interface *<interface>*

interface	Interface name, either FastEthernet 1 or 2 only.

Mode

EXEC: **XSR>**

Sample Output

This sample output displays configuration data of a virtual router on interface *FastEthernet 2*:

```
XSR#show vrrp interface fastethernet 2

Eathernet Interface:       2
Group ID:                  2
State:                     master
Preempt:                   Preempt Enable
Priority:                  15
Adver-int:                 1
Advertise Interval Timer:  1
Authentication Code:       mypass
Virtual IP:                3.3.3.3
Primary IP:                2.2.2.2
Master Router IP:          2.2.2.2
Virtual MAC:               0x00005e005101
BecomeMaster:              2
AdvertiseRcvd:             96
ChecksumErrors:            0
VersionErrrors:            0
PriorityZeroPktsRcvd:      0
PriorityZeroPktsSend:      0
InvalidTypePktsRcvd:       0
UnknownAuthType:           0
AuthTypeErrors:            10
AuthFailures:              0
```

Parameter Description

Fast Ethernet Interface	Interface type and number
Group ID	VRRP group number

State	Master or backup
Preempt	Preempt enabled or not
Preempt-Delay	Preempt delay seconds
Priority	Priority of this group
Adver-int	Advertisement interval
Master Down Timer/ Advertise Interval Timer/ Master Delay Timer	If in backup state, displays the seconds remaining to trigger Master Down Timer or Master Delay Timer; if in master state, displays the seconds remaining to trigger the next advertisement.
Authentication Code	Password
Virtual IP	Virtual IP address
Primary IP	Interface IP address
Master Router IP	Master router IP address
Master-respond-ping	Master-respond-ping enabled or not
Track Interface	Interface being monitored
Virtual MAC	Virtual Mac address
BecomeMaster	Become Master counter
AdvertiseRcvd	Advertisement received packets counter
ChecksumErrors	ChecksumErrors packets counter
VersionErrors	VersionErrrors packets counter
PriorityZeroPktsRcvd	PriorityZeroPktsRcvd counter
PriorityZeroPktsSend	PriorityZeroPktsSend counter
InvalidTypePktsRcvd	InvalidTypePktsRcvd counter
UnknownAuthType	UnknownAuthType packets counter
AuthTypeErrors	AuthTypeErrors packets counter
AuthFailures	AuthFailures packets counter

show vrrp summary

This command displays VRRP summary information on this router.

Syntax

```
show vrrp summary
```

Mode

EXEC: **XSR>**

Sample Output

The following sample output displays VRRP summary data on the XSR:

```
------------------VRRP SUMMARY----------------------------
Maximum number of VRs per router:               4
```

```
Maximum number of virtual addresses per VR:     11
Number of virtual IP address in use:
Fast Ethernet 1   Fast Ethernet 2  Fast Ethernet 3
VR1       1             1                1
VR3       1
VR2       1
------------------------------------------------------------
```

6

Configuring the Border Gateway Protocol

Observing Syntax and Conventions

The CLI command syntax and conventions use the notation described below.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[*x*]	[ ] Square brackets indicate an optional parameter (*italic*)
[x | y | z]	[|] Square brackets with vertical bar indicate a choice of values
{x | y | z}	{ | } Braces with vertical bar indicate a choice of a required value
[x {y | z}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<*xx*>)	*xx* signifies the interface type and number; e.g., **`F1`**, **`G3`**, **`S2/1.0,M57`**. **`F`** indicates a FastEthernet, and **`G`** a GigabitEthernet interface.
Next Mode entries display the CLI prompt after a command is entered.	
Sub-command headings are displayed in ***red***, italicized text.	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

BGP Configuration Commands

The following command subsets define BGP functionality on the XSR, including:

- "BGP Configuration Commands" on page 6-83.
- "Route Map Commands" on page 6-110.
- "BGP Set Commands" on page 6-114.
- "BGP Clear and Show Commands" on page 6-122.
- "BGP Debug Commands" on page 6-132.

router bgp

This command activates a BGP routing process, after which you can configure these additional parameters:

- BGP neighbors

- Networks
- Neighbor parameters
- Routing policies

Syntax

router bgp *autonomous-system*

autonomous-system	The XSR's Autonomous System (AS) number, ranging from 1 to 65,535. The AS number is included in routing updates traded by BGP routers.

Syntax of the "no" Form

The *no* form of this command sets the default parameter - disabled:

no router bgp *autonomous-system*

Mode

Global configuration: **XSR(config)#**

Examples

The following example activates the BGP routing process on a router belonging to AS 100. Note that the XSR acquires Router configuration mode after executing the command:

```
XSR(config)#router bgp 100
XSR(config-router)#
```

The following example displays an error message when you try to activate another BGP process when one is already running. In this example the BGP process was already activated with AS 100 when an attempt was made to activate it again with the AS 11.

```
XSR(config)#router bgp 11
% BGP Already running in AS 100
```

aggregate-address

This command creates an aggregate entry in a BGP routing table which is useful for reducing the number of advertised routes between BGP routers. An aggregate entry in the table is a single summarized route that represents multiple, more specific routes.

At least one of the more specific routes being aggregated must exist in the table for this command to take effect.

Syntax

aggregate-address *address mask* [**as-set**][**summary-only**] [**advertise-map** *map-name*][**attribute-map** *map-name*]

address	The aggregate IP address.
mask	The aggregate IP mask.

as-set	Prevents data loss, including contents of BGP attributes, from more specific routes in the aggregate route. Note that when the contents of those attributes vary within more specific routes, reducing them to the same value within corresponding attributes of the aggregate route can cause routing problems such as loops.
summary-only	Prevents more specific routes that comprise the aggregate route from being advertised.
advertise-map *map-name*	The route map used to select the routes that comprise AS-SET origin communities, ranging from 1 to 199.
attribute-map *map-name*	The route map used to set the attribute of the aggregate route, ranging from 1 to 199.

Syntax of the "no" Form

The *no* form of this command removes the aggregate entry from the table:

no aggregate-address *address mask*

Mode

Router configuration: **XSR(config-router)#**

Default

Disabled

Example

The following example aggregates routes ranging from 192.168.0.0 to 192.168.255.0, each with a mask of 255.255.255.0, into a single aggregate route of 192.168.0.0 with a mask of 255.255.0.0. The optional *summary-only* keyword can be used to direct only the aggregate route be advertised to this router's neighbors. Ommiting the *as-set* option can indicate that all of the routes originate in the same AS and follow the same routing policy, this resulting in no loss of any BGP attribute data within the aggregate.

```
XSR(config)#router bgp 100
XSR(config-router)#aggregate-address 192.168.0.0 255.255.0.0 summary-only
```

auto-summary

This command restores the default behavior of BGP by summarizing redistributed IGP subnets on classful network boundaries. Automatic summarization of IGP subnets reduces the number of routes in the BGP routing table, improving router performance and reducing the amount of bandwidth used by routing traffic between BGP peers.

Syntax

auto-summary

Syntax of the "no" Form

The *no* form of this command removes BGP summarization:

no auto-summary

Mode

Router configuration: **XSR(config-router)#**

Default

Enabled

Example

The following example configures summarization in BGP process *100*:

```
XSR(config)#router bgp 100
XSR(config-router)#auto-summary
```

bgp always-compare-med

This command instructs the XSR to compare the Multi Exit Discriminator (MED) value for paths from neighbors in different ASs. MED is one of the parameters considered by the XSR when selecting the best path. The path with the lowest MED value is chosen when all higher-ranking BGP route selection criteria are the same for all competing paths to the same destination.

Syntax

```
bgp always-compare-med
```

Syntax of the “no” Form

The *no* form of this command removes the MED value:

```
no bgp always-compare-med
```

Mode

Router configuration: **XSR(config-router)#**

Default

The default value for this command is to only compare the MED values for paths from neighbors in the same AS.

Example

The following example sets MED within BGP process *100*:

```
XSR(config)#router bgp 100
XSR(config-router)#bgp always-compare-med
```

bgp bestpath med missing-as-worst

This command specifies that a route with a MED is always considered better than a route without a MED by causing the missing MED attribute to have a value of infinity.

Syntax

```
bgp bestpath med missing-as-worst
```

Syntax of the “no” Form

The *no* form of this command restores the default state, where the missing MED attribute is considered to have a value of zero:

```
no bgp bestpath med missing-as-worst
```

Mode

Router configuration: **`XSR(config-router)#`**

Default

A missing MED attribute is considered to have a value of zero.

Example

This example configures the bgp bestpath med missing-as-worst value within BGP process *100*:

```
XSR(config)#router bgp 100
XSR(config-router)#bgp bestpath med missing-as-worst
```

bgp client-to-client reflection

This command instructs the XSR to reflect routes from a BGP route reflector to clients. When a full IBGP mesh already exists, route reflection is redundant and can be disabled by using the **`no bgp client-to-client reflection`** command.

Syntax

```
bgp client-to-client reflection
```

Syntax of the “no” Form

The *no* form of this command disables the default reflection behavior:

```
no bgp client-to-client reflection
```

Mode

Router configuration: **`XSR(config-router)#`**

Default

Route reflection is enabled.

Example

This example first disables the default reflection setting on this router then restores the default:

```
XSR(config)#router bgp 100
XSR(config-router)#no bgp client-to-client reflection
XSR(config-router)#bgp client-to-client reflection
```

bgp cluster-id

This command sets the cluster identifier for a BGP cluster that contains more than one route reflector. A cluster is comprised of one or more route reflectors and clients of those reflectors. Clusters containing one route reflector are identified by the router identifier of the route reflector.

Syntax

bgp cluster-id *cluster-id*

cluster-id	The cluster of the XSR acting as a route reflector. Valid values are cluster identifiers of up to 4 bytes. Range: 1 to 4294967295 or A.B.C.D (IP address format).

Syntax of the “no” Form

The *no* form of this command resets the cluster identifier to the default:

no bgp cluster-id

Mode

Router configuration: **XSR(config-router)#**

Default

The default value is the router identifier of the route reflector in the cluster.

Example

The following example configures the bgp cluster-id value within the BGP process *600*. The BGP process corresponds to the AS in which the router resides. The cluster ID is configured as *88*. This example configures the cluster ID with two route reflector clients (192.168.1.1, 192.168.1.2).

```
XSR(config)#router bgp 600
XSR(config-router)#bgp cluster-id 88
XSR(config-router)#neighbor 192.168.1.1 remote-as 600
XSR(config-router)#neighbor 192.168.1.1 route-reflector-client
XSR(config-router)#neighbor 192.168.1.2 remote-as 600
XSR(config-router)#neighbor 192.168.1.2 route-reflector-client
```

bgp confederation identifier

This command sets a BGP confederation identifier for a confederation of ASs. A confederation identifier is a valid AS number that represents a confederation comprised of two or more ASs. A confederation appears as a single AS to ASs outside of the confederation.

Syntax

bgp confederation identifier *autonomous-system*

autonomous-system	AS number, ranging from 1 to 65535.

Syntax of the "no" Form

The *no* form of this command removes the confederation identifier:

no bgp confederation identifier

Mode

Router configuration: **XSR(config-router)#**

Example

The following example configures BGP confederation identifier *44* within BGP process *100*:

```
XSR(config)#router bgp 100
XSR(config-router)#bgp confederation identifier 44
```

bgp confederation peers

This command defines ASs belonging to a confederation which is comprised of two or more ASs. A confederation appears as a single AS to ASs outside the confederation.

Syntax

bgp confederation peers *autonomous-system* [*autonomous-system*]

autonomous-system	AS number, ranging from 1 to 65535.

Syntax of the "no" Form

The *no* form of this command deletes the confederation Ss:

no bgp confederation peers *autonomous-system*
[*autonomous-system*][*autonomous-system*...]

Mode

Router configuration: **XSR(config-router)#**

Example

The following example configures the BGP confederation peers value within BGP process *100*. The ASs assigned to the confederation using this command are *600, 700,* and *800.* Confederation *44* is configured using the **bgp confederation identifier** command. The AS *100* to which this router belongs is also a member of confederation 44.

```
XSR(config)#router bgp 100
XSR(config-router)#bgp confederation identifier 44
XSR(config-router)#bgp confederation peers 600 700 800
```

bgp dampening

This command enables BGP route dampening to minimize propagation of flapping routes (repeatedly available/unavailable) across the network. Each time a route flaps, a penalty value of 1024 is assigned to that route.

Syntax

bgp dampening [*half-life* | *reuse* | *suppress* | *suppress-max*][**route-map** *route-map-number*]

half-life	Interval after which the route's penalty becomes half its value, ranging from 1 to 45 minutes.
reuse	How *low* a route's penalty must become before the route becomes eligible for use again after being suppressed, ranging from 1 to 20000.
suppress	How *high* a route's penalty must become before the route is suppressed, ranging from 1 to 20000.
suppress-max	Peak interval a route can be suppressed regardless of how unstable it is. Range: 1 to 255 minutes.
route-map-number	Route map number applied to dampened routes, ranging from 1 to 199.

Syntax of the "no" Form

The *no* form of this command disables BGP dampening:

```
no bgp dampening
```

Mode

Router configuration: **XSR(config-router)#**

Defaults

- *Half-life* - 15 minutes
- *Reuse* - 750
- *Suppress* - 2000
- *Suppress-max* - 60 minutes
- Disabled.

Example

The following example enables route flap dampening:

```
XSR(config)#router bgp 100
XSR(config)#bgp dampening
```

bgp default local-preference

This command changes the default local preference value. The path with the highest local preference value is preferred over competing paths to the same destination provided that all higher-ranking route selection criteria of those paths are the same. The local preference value for the path is sent to all routers and access servers in the local AS.

Syntax

bgp default local-preference *value*

value	Local preference value, ranging from 0 to 4294967295.

Syntax of the "no" Form

The *no* form of this command reverts to the local preference default:

no bgp default local-preference

Mode

Router configuration: **XSR(config-router)#**

Default

100

Example

This example configures the BGP default local-preference of 300 for BGP process *100*. This setting indicates that all routes this router advertises to its internal BGP neighbors will have a local preference of 300.

```
XSR(config)#router bgp 100
XSR(config-router)#bgp default local-preference 300
```

distance bgp

This command sets the BGP route preference - administrative distance - for its external and internal routes submitted to the routing table.

Syntax

distance bgp *external internal*

external	The administrative distance for external BGP routes - those learned from neighbors external to the AS - ranging from 1 to 240.
internal	The administrative distance for internal BGP routes - those learned from neighbors within the same AS - ranging from 1 to 240.

Syntax of the "no" Form

The *no* form of the command removes the configured value:

no distance bgp

Defaults

- External: 20
- Internal: 200

Mode

Router configuration: **XSR(config-router)#**

Example

This example sets BGP external and internal administrative distances to *50* and *150*, respectively:

```
XSR#config terminal
XSR(config)#router bgp 100
XSR(config-router)#distance bgp 50 150
```

neighbor advertisement-interval

This command sets the minimum interval that a router waits between sending BGP routing updates to its neighbor. Before entering this command, a neighbor or peer group must be identified by means of the neighbor remote-as or neighbor peer-group command. Configuring a minimum interval of zero means that there is no delay in sending BGP routing updates to its neighbor.

Syntax

neighbor {*ip-address* | *peer-group-name*} **advertisement-interval** *seconds*

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.
seconds	Minimum interval, ranging from 0 to 600 seconds.

Syntax of the "no" Form

The *no* form returns to the advertisement interval default:

no neighbor {*ip-address* | *peer-group-name*}
advertisement-interval *seconds*

Mode

Router configuration: **XSR(config-router)#**

Default

- External peers: 30 seconds
- Internal peers: 5 seconds

Example

The following example sets the neighbor advertisement-interval value within BGP process *100*. Note that the **neighbor remote-as** command must be executed before this command can be entered. In the example, the router on which the configuration occurs resides in AS *100*. Neighbor 192.168.1.1 resides in AS 101. The default update interval between these peers has been changed from 30 to 90 seconds.

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 advertisement-interval 90
```

neighbor default-originate

This command sends the route 0.0.0.0 to the BGP neighbor of the router that this command is entered on so that it can be used as the default route. Before entering this command, a neighbor or peer group must be identified by means of the **neighbor remote-as** or **neighbor peer-group** commands.

Syntax

neighbor {*ip-address* | *peer-group-name*} **default-originate**

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.

Syntax of the "no" Form

The *no* form of this command returns to the default value:

no neighbor {*ip-address* | *peer-group-name*} **default-originate**

Mode

Router configuration: **XSR(config-router)#**

Default

Disabled

Example

This example sets the local router to unconditionally inject route *0.0.0.0* to neighbor *192.168.1.1*:

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 default-originate
```

neighbor distribute-list

This command distributes the information specified in an access-list to a BGP neighbor. Before entering this command, a neighbor or peer group must be identified by means of the **neighbor remote-as** or **neighbor peer-group** command. Also, the prefix-based ACL must be configured.

Note: Perform a **clear ip bgp neighbor <IP address>** whenever this command is changed.

Syntax

neighbor {*ip-address* | *peer-group-name*} **distribute-list** *access-list* {**in** | **out**}

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.
access-list	ACL, ranging from 1 to 199.
in	ACL applied to inbound routes.
out	ACL applied to outbound routes.

Syntax of the "no" Form

The *no* form of this command removes the ACL-linked neighbor:

no neighbor {*ip-address* | *peer-group-name*} **distribute-list** *access-list* {**in** | **out**}

Mode

Router configuration: **XSR(config-router)#**

Default

No access list applied

Example

This example applies access-list 1 to incoming advertisements from neighbor *192.168.1.1*. Only routes which match *10.0.0.0/8*, *11.0.0.0/8* or *12.0.0.0/8* prefixes will be accepted from the neighbor.

```
XSR(config)#access-list 1 permit 10.0.0.0 255.0.0.0
XSR(config)#access-list 1 permit 11.0.0.0 255.0.0.0
XSR(config)#access-list 1 permit 12.0.0.0 255.0.0.0
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 distribute-list 1 in
```

neighbor ebgp-multihop

This command connects the BGP neighbors on networks that are not directly-connected to the network of the router that this command is entered on. Before entering this command, a neighbor or peer group must be identified by means of the **neighbor remote-as** or **neighbor peer-group** command.

Syntax

neighbor {*ip-address* | *peer-group-name*} **ebgp-multihop**

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.

Syntax of the "no" Form

The *no* form of this command removes the specified neighbor:

no neighbor {*ip-address* | *peer-group-name*} **ebgp-multihop**

Mode

Router configuration: **XSR(config-router)#**

Default

Not enabled

Example

The following example allows connections to or from neighbor 192.168.1.1, which resides on a network that is not directly connected:

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 ebgp-multihop
```

neighbor filter-list

This command sets up a BGP filter based on AS path. Before entering this command, a neighbor or peer group must be identified by means of the **neighbor remote-as** or **neighbor peer-group** command. Also, the AS path-based access list must be configured.

Note: Perform a **clear ip bgp neighbor <IP address>** whenever this command is changed.

Syntax

neighbor {*ip-address* | *peer-group-name*} **filter-list** *filter-list* {**in** | **out** | **weight** *value*}

ip-address	Neighbor's IP address.

peer-group-name	BGP peer group by name. Range: 1 to 64 characters.
filter-list	Identifies the AS path access list. Range is 1-199.
in	Filter list is applied to inbound routes.
out	Filter list is applied to outbound routes.
weight	Assigns a weight to all routes matching the filter list.
value	Weight range from 0 to 65535.

Syntax of the "no" Form

The *no* form of this command removes the specified neighbor:

no neighbor {*ip-address* | *peer-group-name*} **filter-list** *filter-list*

Mode

Router configuration: **XSR(config-router)#**

Example

This example applies filter list 1 to incoming advertisements from neighbor 192.168.1.1. Only routes which start with AS path 200 and end with AS path 500 will be accepted from the neighbor.

```
XSR(config)#ip as-path access-list 1 permit "^200 .* 500$"
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 filter-list 1 in
```

neighbor maximum-prefix

This command controls the number of prefixes received from a particular neighbor. When the maximum number of prefixes is exceeded, a CEASE message is sent and the connection is cleared. To reactivate the session, enter **clear ip bgp <IP address>**. If the number of prefixes is set to zero, no prefixes will be accepted from the neighbor.

Syntax

neighbor {*ip-address* | *peer-group-name*} **maximum-prefix** *value* [*threshold*][**warning-only**]

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.
value	Maximum number of prefixes that can be received from a neighbor, ranging from 1 to 4,294,967,295.
threshold	The threshold value - percentage of maximum - at which a warning is generated, ranging from 1 to 100 prefixes.
warning-only	When the maximum number of prefixes is reached the XSR generates a warning message instead of terminating the peering session.

Syntax of the "no" Form

The *no* form of this command removes the specified neighbor:

no neighbor {*ip-address* | *peer-group-name*} **maximum-prefix** *value* [**threshold**] [**warning-only**]

Mode

Router configuration: **XSR(config-router)#**

Defaults

- No restriction on the number of prefixes.
- Threshold: 75 prefixes

Example

The following example sets the maximum number of prefixes allowed from the neighbor at *192.168.1.1* to *10000*:

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 maximum-prefix 10000
```

neighbor next-hop-self

This command disables automatic next-hop selection. Updates meant for the specified system or peer group are forced to advertise this router as the next hop.

Syntax

neighbor {*ip-address* | *peer-group-name*} **next-hop-self**

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.

Syntax of the "no" Form

The *no* form of this command returns to the default value:

no neighbor {*ip-address* | *peer-group-name*} **next-hop-self**

Mode

Router configuration: **XSR(config-router)#**

Default

Next hop selection is performed automatically by BGP.

Example

The following example sets the router at 192.168.1.1 as the next hop:

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 next-hop-self
```

neighbor password

This command sets a password for Message Digest 5 (MD5) authentication on the TCP connection between the XSR that this command is entered on and a BGP neighbor. The same password must be configured on both routers. When a password is configured for a neighbor, the existing session is replaced by a new session.

Syntax

neighbor {*ip-address* | *peer-group-name*} **password** *password-value*

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.
password-value	Alphanumeric password. Range is 1-30 characters.

Syntax of the "no" Form

This command's *no* form removes the password for the specified router:

no neighbor {*ip-address* | *peer-group-name*} **password** *password-value*

Mode

Router configuration: **XSR(config-router)#**

Default

No authentication

Example

The following example adds a password for the specified router:

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 password 123456
```

neighbor peer-group

This command creates a BGP peer group and assigns a BGP neighbor to a peer group.

Syntax

neighbor {*ip-address* | *peer-group-name*} **peer-group** [*peer-group-name*]

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.

Syntax of the "no" Form

The *no* form of this command removes the specifed neighbor peer group:

no neighbor {*ip-address* | *peer-group-name*} **peer-group** [*peer-group-name*]

Mode

Router configuration: **XSR(config-router)#**

Example

The following example creates peer group *ExternalGroup* and assigns neighbor 192.168.1.1 to peer group *ExternalGroup*:

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor ExternalGroup peer-group
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 peer-group ExternalGroup
```

neighbor remote-as

This command adds an entry to the BGP neighbor table. BGP requires manual neighbor configuration. The configuration of neighbors on both of the neighboring BGP routers allows a BGP session to be set up between the routers and allows the exchange of BGP update messages.

For external BGP neighbors, the IP address specified is that of the neighbor interface to the shared subnet between routers (unless *ebgp-multihop* is enabled). For internal BGP neighbors, the neighbor IP address is any reachable IP address from the router.

Syntax

neighbor {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system*

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.
autonomous-system	AS by number, ranging from 1 to 65535.

Syntax of the "no" Form

The *no* form of this command removes the specified entry from the table:

no neighbor {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system*

Mode

Router configuration: **XSR(config-router)#**

Example

The following example configures two neighbors. Neighbor *192.168.1.1* is an external neighbor since the AS number of *101* differs from the AS number for the router *100*. Neighbor *192.168.2.1* is an internal neighbor since it resides in the same AS *100*.

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.2.1 remote-as 100
```

neighbor route-map

This command applies a route map to routes that enter from or exit out of a BGP neighbor or peer group. The route map must be configured first.

Note: Perform a **clear ip bgp neighbor <IP address>** whenever this command is changed.

Syntax

neighbor {*ip-address* | *peer-group-name*} **route-map** *route-map#* {**in** | **out**}

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.
route-map#	Identifies the route map number. Range: 1-199.
in	Route map is applied to inbound routes.
out	Route map is applied to outbound routes.

Syntax of the "no" Form

The *no* form of this command deletes the specified neighbor's route map:

no neighbor {*ip-address* | *peer-group-name*} **route-map** *route-map#* {**in** | **out**}

Mode

Router configuration: **XSR(config-router)#**

Example

The following example adds a neighbor route map:

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 route-map 1 in
```

neighbor route-reflector-client

This command establishes the router that this command was entered on as a BGP route reflector. This command also identifies the specified neighbor router as the client of the BGP route reflector. Neighbors configured with this command are members of the client group and the remaining internal BGP peers are members of the non-client group for the router reflector.

Syntax

neighbor {*ip-address* | *peer-group-name*} **route-reflector-client**

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.

Syntax of the "no" Form

The *no* form of this command removes a neighbor's route reflector:

no neighbor {*ip-address* | *peer-group-name*} **route-reflector-client**

Mode

Router configuration: **XSR(config-router)#**

Example

The following example sets a neighbor's reoute reflector:

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 route-reflector-client
```

neighbor send-community

This command instructs the system to send a community attributed to a BGP neighbor.

Syntax

neighbor {*ip-address* | *peer-group-name*} **send-community**

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.

Syntax of the "no" Form

The *no* form of this command removes a neighbor's community:

no neighbor {*ip-address* | *peer-group-name*} **send-community**

Mode

Router configuration: **XSR(config-router)#**

Example

The following example sets a neighbor's community:

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 send-community
```

neighbor shutdown

This command disables a neighbor or peer-group.

Syntax

neighbor {*ip-address* | *peer-group-name*} **shutdown**

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.

Syntax of the "no" Form

The *no* form of this command returns to the command default:

no neighbor {*ip-address* | *peer-group-name*} **shutdown**

Mode

Router configuration: **XSR(config-router)#**

Default

No change is made to status of BGP neighbor or peer group.

Example

This example disables any active session for neighbor 192.168.1.1:

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 shutdown
```

neighbor soft-reconfiguration inbound

This command instructs the system to store updates as they are received. Updates are required to be stored in order to perform inbound soft reconfiguration.

Syntax

neighbor {*ip-address* | *peer-group-name*} **soft-reconfiguration inbound**

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.

Syntax of the "no" Form

The *no* form of this command returns to the command default:

no neighbor {*ip-address* | *peer-group-name*} **soft-reconfiguration inbound**

Mode

Router configuration: **XSR(config-router)#**

Default

No soft reconfiguration is done.

Example

The following example configures soft reconfiguration on the router:

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 soft-reconfiguration inbound
```

neighbor timers

This command changes the values of BGP timers for a peer or peer group. When a session is started, BGP negotiates the hold-time with the neighbor, selecting the smaller value. The keep-alive timer is then set based on the negotiated hold-time and the configured keep-alive interval. By default, the keep-alive timer is set to 30 seconds and the hold-time timer set to 90 seconds. This 1 to 3 ratio is strictly maintained between the timers.

Note: Perform a **clear ip bgp neighbor <IP address>** whenever this command is changed.

The timers configured for a specific neighbor or peer group override the timers configured for all BGP neighbors using the **timers bgp** command.

Syntax

neighbor {*ip-address* | *peer-group-name*} **timers** *keep-alive*

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group's name, ranging from 1 to 64 characters.
keep-alive	Keep-alive interval, ranging from 0 to 4,294,967,296 seconds. A keep-alive of zero indicates no keep-alives are sent between neighbors so the peer session will not time out.

Syntax of the "no" Form

The *no* form of this command returns to the command default:

no neighbor {*ip-address* | *peer-group-name*} **timers** *keep-alive*

Default

Keep-alive: 30 seconds

Mode

Router configuration: **XSR(config-router)#**

Example

This example sets the peer keep-alive to *10* seconds and, subsequently, the hold-time to *30* seconds:

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 1.1.1.1 timers 10
```

neighbor update-source

This command specifies the source IP address used when communicating with a BGP neighbor. A loopback interface is typically used with this command.

Syntax

neighbor {*ip-address* | *peer-group-name*} **update-source** *interface*

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.
interface	Identifies the interface to be used as the source.

Syntax of the "no" Form

The *no* form of this command removes a neighbor's update source:

no neighbor {*ip-address* | *peer-group-name*} **update-source** *interface*

Mode

Router configuration: **XSR(config-router)#**

Default

Best outbound interface.

Example

The following example sources BGP TCP connections for the specified neighbor with the IP address of the loopback interface rather than the best local address:

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 update-source loopback 0
```

neighbor weight

This command specifies a weight value for a connection to a neighbor or a BGP peer group.

Note: Perform a **clear ip bgp neighbor <IP address>** whenever this command is changed.

Syntax

neighbor {*ip-address* | *peer-group-name*} **weight** *value*

ip-address	Neighbor's IP address.
peer-group-name	BGP peer group by name. Range: 1 to 64 characters.
value	Assigns a weight for all routes learned from this neighbor, ranging from 0 to 65535.

Syntax of the "no" Form

The *no* form of this command removes a neighbor's weight:

no neighbor {*ip-address* | *peer-group-name*} **weight** *value*

Mode

Router configuration: **XSR(config-router)#**

Example

The following example sets the specified neighbor's weight to *100*:

```
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 weight 100
```

ip as-path access-list

This command creates an *as-path* filter list which can be applied to filter inbound and outbound BGP updates. The *as-path* variable in the BGP routing update message is examined against a required parameter of this command, which represents AS numbers identified by means of a regular expression. Multiple regular expressions can be configured under a particular as-path filter list.

Note: Perform a **clear ip bgp** whenever this command is changed.

Syntax

ip as-path access-list *access-list-number* {**permit** | **deny**} *as-regular-expression*

access-list-number	Identifies the access list by number. Range is 1 to 199.
permit	Instructs XSR to permit access to paths matching specified conditions.

deny	Instructs XSR to deny access to paths matching specified conditions.
as-regular-expression	Identifies an AS in the access list by means of the regular expression.

Syntax of the "no" Form

The *no* form of this command removes the configured filter list:

no ip as-path access-list *access-list-number*

Mode

Global configuration: **XSR(config)#**

Example

The following example configures the *IP as-path* access-list value in the context of configuring a route map and performing a match using the **match as-path** command.

The as-path access list is *33*, ends with a regular expression *".* 640 .*"* and is referenced in the match as-path command, which in turn is configured inside of the route map *33*. This means that a match occurs if the *as-path* variable in a BGP update message contains AS *640*.

```
XSR(config)#ip as-path access-list 33 permit ".* 640 .*"
XSR(config)#route-map 33 permit 1
XSR(config-route-map)#match as-path 33
XSR(config-route-map)#set local-preference 300
```

ip community-list

This command defines a community list that filters on the BGP *COMMUNITY* attribute. The community list you define typically is referenced by the **match community** command, which includes a route map that implements routing policies based on community attributes. Multiple community attributes can be configured for a particular community list.

Note: Perform a **clear ip bgp neighbor** whenever this command is changed.

Syntax

ip community-list *community-list-number* {**permit** | **deny**} *community-number*

community-list-number	Community list number (standard), ranging from 1 to 199.
permit	XSR permits access to community lists matching conditions you specifiy.
deny	XSR denies access to community lists matching conditions. you specify.

community-number	Community number as it was defined for this router via the **set community** command. Valid values are: • *Range*: 1 to 4,294,967,200. • *aa:nn:* AS number, Community number. • *internet:* the Internet community. • *no-export:* the community route will not be advertised to an EBGP peer. • *no-advertise:* the route will not be advertised to any peer.

Syntax of the "no" Form

The *no* form of this command removes the commmunity list number:

no community-list *community-list-number*

Mode

Global configuration: **XSR(config)#**

Example

This example configures IP community list *88*. The community numbers specified in the list are *2000*, *3000*, and *4000* in the first, second, and third instance of the command, respectively. This list can be referenced within the **match community** command that is part of a route map controlling BGP routing based on the *community* attribute. The match will seek updates that include community numbers *2000*, *3000*, or *4000*.

```
XSR(config)#ip community-list 88 permit 2000
XSR(config)#ip community-list 88 permit 3000
XSR(config)#ip community-list 88 permit 4000
```

network

This command specifies the list of networks for the BGP routing process. Networks can be learned from *connected routes* or via *dynamic routing*. The BGP process must be notified about the networks it will route which con occurs via manual injection of routes into the BGP process with the **network** command. Routes originated by BGP via the **network** command have their origin code set to IGP.

Network numbers that are injected into BGP by means of the **network** command must already exist in the IP routing table on the router as static, directly-connected, or dynamically-derived routes. If network numbers do not already exist, they will not be placed into the BGP table, even though they will appear in the router's configuration.

Syntax

network *network-number* [*mask network-mask*]

network-number	Network that BGP advertises.
mask	Used when a network-mask is explicitly specified for the network-number. Without the network-mask being specified, a default classful mask is assumed.

network-mask	The mask associated with the network-number for which the BGP process routes. It is specified when the *network-number* represents a subnet as opposed to a classful network.

Syntax of the "no" Form

The *no* form removes the network from the routing table:

no network *network-number* [**mask** *network-mask*]

Mode

Router configuration: **XSR(config-router)#**

Example

The following example configures a network with and without the optional mask keyword. In the optional mask statement, the network-number represents a subnet of class B network *172.17.0.0.* A default Class C network mask is assumed for the network *192.168.1.0* in the configuration statement without the optional parameters.

```
XSR(config)#router bgp 100
XSR(config-router)#network 172.17.151.0 mask 255.255.255.0
XSR(config-router)#network 192.168.1.0
```

redistribute

This command redistributes routes from a protocol into the BGP. Redistributed routes can be learned from dynamic routing (OSPF, RIP), static routes, and connected routes.

Redistributed routes can have their path attributes set in BGP by the **route-map** command. By default, redistributed static routes have their origin code set to *incomplete* unless otherwise configured by **route-map**.

Syntax

redistribute {**ospf** | **rip** | **static** | **connected**} [**metric** *metric-value* | **route-map** *route-map-name*]

ospf	OSFP routes.
rip	RIP routes.
static	Static routes.
connected	Connected routes.
metric-value	Metric for redistributed routes. Range: 0-4294967295.
route-map-name	Route map applied to redistributed routes, ranging from 1 to 199.

Syntax of the "no" Form

The *no* form of this command returns to the command default:

no redistribute {**ospf** | **rip** | **static** | **connected**}

Mode

Router configuration: **`XSR(config-router)#`**

Default

Redistribution is not enabled.

Example

The following example redistributes static routes into BGP:

```
XSR(config)#router bgp 100
XSR(config-router)#redistribute static
```

synchronization

This command synchronizes BGP with the IGP in the AS. You should synchronize BGP with IGP if there are routers in the AS that are not BGP routers.

Syntax

```
synchronization
```

Syntax of the “no” Form

The *no* form of this command disables synchronization:

```
no synchronization
```

Mode

Router configuration: **`XSR(config-router)#`**

Default

Enabled

Example

The following example disables synchronization:

```
XSR(config)#router bgp 100
XSR(config-router)#no synchronization
```

timers bgp

This command resets BGP timers. When a session is started on a router, BGP negotiates *hold-time* with the neighbor and selects the smaller value. The *keepalive* timer is then set based on the *negotiated* hold-time and the *configured* keepalive period. By default, the keepalive timer is set at 60 seconds and the holdtime timer is set at 180 seconds. It is recommended you maintain this 1 to 3 ratio between the timers.

Syntax

timers bgp *keep-alive*

keep-alive	Keepalive interval. A keep alive of zero indicates no keepalives are sent between neighbors so the peer session will not time out. Range: 0 - 4294967296 seconds.

Syntax of the "no" Form

The *no* form of this command deletes the timers value:

no timers bgp

Mode

Router configuration: **XSR(config-router)#**

Defaults

- Keepalive timer: 30 seconds
- Holdtime timer: 90 seconds

Example

The following example sets the hold-timer interval to 30 seconds:

```
XSR(config)#router bgp 100
XSR(config-router)#timers bgp 30
```

Route Map Commands

Route maps are comprised of sets of match and set commands. Match commands define the match criteria for route maps. Routes that match all defined match criteria are processed via set commands and those that do not match all of the defined match criteria in the route map are ignored.

match as-path

This command matches the values of the *as_path* variable in BGP routing update messages to the values of AS numbers identified through the AS-path access list.

A route must match at least one match statement of a **route-map** command. If a route does not match any match statements, the route is not advertised on outbound route maps and is not accepted on inbound route maps.

Syntax

match as-path *path-list-number*

path-list-number	AS-path access list to match, ranging from 1 to 199.

Syntax of the "no" Form

The *no* form of this command removes the patch list number:

no match as-path *path-list-number*

Mode

Route-map configuration: **XSR(config-route-map)#**

Example

This example sets the *match as-path* in the context of configuring a route map and as-path ACL *33*.

```
XSR(config)#route-map 1 permit 1
XSR(config-route-map)#match as-path 33
XSR(config-route-map)#set local-preference 300
XSR(config-route-map)#exit
XSR(config)#ip as-path access-list 33 permit ".* 550 .*"
```

Route map 1 is configured with the optional *permit* keyword and sequence number *1*. If these values are omitted, a route map will default to the *permit* keyword and sequence number 10.

After route map *1* is defined via the **route-map** command, you enter the **match as-path** command which references as-path access list 33 - the last configuration statement in the example. AS-path access list 33 ends with a regular expression *".* 550 .*"*, indicating a match will occur if the *as_path* variable in a BGP update message contains AS number *550*.

If a match occurs, then the **set local-preference** command sets the local preference attribute for the matching BGP updates to *300*, overriding the default value of *100*. A route flagged with a higher local preference value is more preferable to a route with a lower local preference. Consequently, the routes passing through AS *550* become more preferable to other routes for the same destinations.

match community-list

This command matches the community attribute in a BGP routing update message with the values of the community attribute identified through the community access list.

A route must match at least one match statement of a **route-map** command. If a route does not match any match statements, the route is not advertised on outbound route maps and is not accepted on inbound route maps.

Syntax

match community-list *community-list-number*

community-list-number	Community ACL to match by number, ranging from 1 to 199.

Syntax of the "no" Form

The *no* form of this command removes the community list number:

no match community-list *community-list-number*

Mode

Route-map configuration: **XSRA(config-route-map)#**

Default

No match based on community list

Example

The following example configures the match community value in the context of configuring a route map named *1* and community list *77* on *XSRA* and *XSRB*:

Router A configuration:

```
XSRA(config)#route-map 1 permit 1
XSRA(config-route-map)#match community 77
XSRA(config-route-map)#set local-preference 500
XSRA(config-route-map)#exit
XSRA(config)#ip community-list 77 permit 300:22
```

Router B configuration:

```
XSRB(config)#route-map 1 permit 1
XSRB(config-route-map)#match community 77
XSRB(config-route-map)#set local-preference 200
XSRB(config-route-map)#exit
XSRB(config)#ip community-list 77 permit 300:22
```

XSRA and *XSRB* are border routers within the same AS. The community is identified by name 300:22. The numeric format *aa:nn*, where *aa* and *nn* represent two-byte numbers, is one of the allowable formats for community names. BGP updates matching community name *300:22* are assigned a higher local preference on *XSRA* (500) than on *XSRB* (200). This makes *XSRA* the preferable exit point from this AS for the networks that have been grouped under the community name 300:22. Use the **set community** command to assign community names.

match metric

This command matches the MED attribute in a BGP routing update message. A route must match at least one match statement of a **route-map** command. If a route does not match any match statements, the route is not advertised on outbound route maps and is not accepted on inbound route maps.

Syntax

match metric *metric-value*

metric-value	MED value to match, ranging from 0 to 2147483647.

Syntax of the “no” Form

The *no* form of this command removes the match metric value:

no match metric *metric-value*

Mode

Route-map configuration: **XSR(config-route-map)#**

Example

The following example sets the match metric to *300*:

```
XSR(config)#route-map 1 permit 1
XSR(config-route-map)#match metric 300
```

match ip address

This command matches IP addresses in a BGP routing update message. A route must match at least one match statement of a **route-map command**. If this is not done, the route is not advertised on outbound route maps and is not accepted on inbound route maps.

Syntax

match ip address *access-list-number*

access-list-number	The ACL to match, ranging from 1 to 199.

Syntax of the "no" Form

The *no* form of this command removes the match IP address value:

no match ip address *access-list-number*

Mode

Route-map configuration: **XSR(config-route-map)#**

Default

No matching based on IP prefix.

Example

The following example sets the matching IP address to *10*:

```
XSR(config)#access-list 10 permit 10.0.0.0 255.0.0.0
XSR(config)#route-map 1 permit 1
XSR(config-route-map)#match ip address 10
```

match ip next-hop

This command matches the value of the next hop attribute in a BGP routing update message against an ACL specified by the command. A route must match at least one match statement of a **route-map command**. If a route does not match any match statements, it is not advertised on outbound route maps and is not accepted on inbound route maps.

Syntax

match ip next-hop *access-list-number*

access-list-number	The ACL to match, ranging from 1 to 199.

Syntax of the "no" Form

The *no* form of this command removes the match next hop value:

no match ip next-hop *access-list-number*

Mode

Route-map configuration: **XSR(config-route-map)#**

Default

No matching based on IP next hop.

Example

The following example sets the matching IP next hop to 10:

```
XSR(config)#access-list 10 permit 1.2.3.4
XSR(config)#route-map 1 permit 1
XSR(config-route-map)#match ip next-hop 10
```

BGP Set Commands

Route maps are comprised of sets of match and set commands. Match commands define the match criteria for route maps. Routes that match all defined match criteria are processed via set commands and those that do not match all of the defined match criteria in the route map are ignored.

set as-path

This command increases the length of the AS-path attribute for the BGP routing update messages that meet the match conditions specified within a route map.

The length of the AS path attribute influences the BGP route selection process for destinations that can be reached by means of multiple paths. AS path length is the only global BGP metric that you can use to influence best-path selection. A BGP speaker can influence the best path selection by a peer by varying the length of the AS path.

If you do not set local preference or weight, AS path length determines which of multiple routes are selected. Routes with longer autonomous system paths are preferred. To prefer a path, you can pad the autonomous system path by prepending extra autonomous system numbers.

Syntax

set as-path prepend *as-path-string*

prepend	Instructs the system to attach the *as-path-string* value to the AS path of the route that matches the route map.

as-path-string	The AS path list which will be prepended to the AS path attribute of the route that matches the route map. The *as-path list* represents one or more valid AS numbers that are specified as an integer between 1 and 65535.

Syntax of the "no" Form

The *no* form of this command removes the AS path value:

```
no set as-path
```

Mode

Route-map configuration: **XSR(config-route-map)#**

Example

The following example configures the *as-path* value in the context of configuring a route map and the **match** command. The **match as-path** command references AS-path access list *37* which identifies the BGP routing updates to which the **set as-path** command will apply.

In this case, match clause *".*"* will match all routes. Relevant updates will have one instance of the AS number *100* prepended into their AS path variable. Assuming that all of the BGP route selection criteria remain the same, the routes with the fewest AS numbers in the AS path variable will be chosen as the best routes to the identified destinations. If more than one AS path is to be prepended, then the string should be surrounded by quotes.

```
XSR(config)#ip as-path access-list 37 permit ".*"
XSR(config)#route-map 1 permit 1
XSR(config-route-map)#match as-path 37
XSR(config-route-map)#set as-path prepend 100
XSR(config-route-map)#set as-path prepend "100 100"
```

set community

This command specifies the community attribute in a BGP routing update message. Be sure that a match clause has been specified.

A community is a group of destinations which share the community attribute. A BGP speaker can use the community attribute to control which routing data it accepts or distributes to neighbors. A BGP speaker can append the community attribute to routes it receives that do not already have the attribute.

Syntax

set community {*community-number* | *aa:nn* | *additive* | **internet** | **local-AS** | **no-advertise** | **no-export** | **none**}

community-number	The community number. Range: 1 to 4,294,967,295.
aa:nn	Community number in the format aa:nn where **aa** identifies the AS and **nn** the community within the AS. Range: 1 to 65,535.
additive	Adds the community to existing communities.
internet	Established Internet community.

local-AS	Established community which specifies that routes containing this value should not be advertised to external BGP peers.
no-advertise	Established community which specifies that routes containing this value should not be advertised to any other BGP peers (internal or external).
no-export	Established community which specifies that routes containing this value should not be advertised outside a BGP confederation boundary.
none	Removes any existing communities.

Syntax of the “no” Form

The *no* form of this command removes the set community value:

```
no set community
```

Mode

Route-map configuration: **XSR(config-route-map)#**

Example

The following example configures the *set community* value in the context of configuring route map *1* and the neighbor send community value:

```
XSR(config)#ip access-list 37 permit 10.0.0.0 255.0.0.0
XSR(config)#route-map 1 permit 1
XSR(config-route-map)#match ip address 37
XSR(config-route-map)#set community 500:10
XSR(config-route-map)#exit
XSR(config)#route-map 1 permit 2
XSR(config-route-map)#set community none
XSR(config-route-map)#exit
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 send-community
XSR(config-router)#neighbor 192.168.1.1 route-map 1 out
```

Route map 1 is applied to the outgoing BGP updates between this router and its peering neighbor identified by IP address *192.168.1.1* in AS *101*. The first instance of route map 1 matches the destinations in the BGP updates against the criteria specified in ACL *37 (10.0.0.0/8)*. If there is not a match, the second instance of route map *1* is invoked, which matches on all remaining routes and removes any community attributes. This means that routes matching ACL *37* criteria will have a community attribute set to *500:10*, but all of the other routes advertised to *192.168.1.1* will not.

The BGP peer *192.168.1.1* will then have the option to apply a desired routing policy to all routes arriving from this router with the community attribute set to *500:10*.

set dampening

This command configures route flap dampening, a mechanism to combat network overhead which arises from the proliferation of uncontrolled disconnecting/reconnecting networks.

With route dampening, you can address these problem routes as follows:

- The XSR penalizes a route marked as unstable with a value of 1024 each time it fails. If penalties accrue beyond the *suppress* threshold you set, the route is no longer advertised.
- The XSR permits suppressed routes to rejoin the BGP routing table when their penalties drop below the threshold.
- After a route assumes a penalty, the XSR cuts the penalty in half each time a *half-life* interval you configure elapses.
- When penalties drop below the configurable *reuse* value, the XSR frees the route, re-inserting it into the BGP routing table.
- The XSR does not suppress routes indefinitely. You can set the *max-suppress* value and fix the maximum period a route can be suppressed before it is advertised again.

Syntax

set dampening *half-life | reuse | suppress | suppress-max*

half-life	Interval after which the route's penalty becomes half its value, ranging from 1 to 45 minutes.
reuse	Specifies how low a route's penalty must become before the route becomes eligible for use again after being suppressed, ranging from 1 to 20,000 seconds.
suppress	Specifies how high a route's penalty must become before the route is suppressed, ranging from 1 to 20,000.
suppress-max	Specifies that maximum interval in minutes that a route can be suppressed regardless of how unstable it is, ranging from 1 to 20,000 minutes.

Syntax of the "no" Form

The *no* form of this command removes route dampening:

```
no set dampening
```

Mode

Route-map configuration: **XSR(config-route-map)#**

Defaults

- Half-life: *15 minutes*
- Reuse: *750 seconds*
- Suppress: *2000*
- Suppress-max: *60 minutes - four* times the half-life value.

Example

This example displays the use of the set dampening for IP prefix *10.0.0.0* for BGP process *100*:

```
XSR(config)#ip access-list 10 permit 10.0.0.0 255.0.0.0
XSR(config)#route-map 1 permit 1
XSR(config-route-map)#match ip address 10
XSR(config-route-map)#set 30 1500 10000 120
```

```
XSR(config)#router bgp 100
XSR(config)#bgp dampening route-map 1
```

set ip next-hop

This command specifies where to output packets that pass a match clause of a route map for policy routing. It modifies the value of the next hop attribute in a BGP routing update message.

The next-hop attribute identifies the next hop to reach a route. Next-hop for an EBGP session is the IP address of the BGP neighbor that announced the route. Next-hop for IBGP sessions is either the BGP neighbor that announced the route (for routes that originate inside the AS) or the BGP neighbor from which the route was learned (for routes injected into the AS via EBGP).

Syntax

set ip next-hop *value*

value	The next hop IP address.

Syntax of the “no” Form

The *no* form of this command removes the next hop value:

no set ip next-hop *value*

Mode

Route-map configuration: **XSR(config-route-map)#**

Example

The following example sets the IP next hop attribute in the BGP update which matches *10.0.0.0 255.0.0.0* to *1.2.3.4*:

```
XSR(config)#access-list 10 permit 10.0.0.0 255.0.0.0
XSR(config)#route-map 1 permit 1
XSR(config-route-map)#match ip address 10
XSR(config-route-map)#set ip next-hop 1.2.3.4
```

set local-preference

This command modifies the value of the local preference attribute in a BGP routing update message. This parameter impacts the BGP route selection process for traffic leaving an AS. Be sure that a match clause has been specified.

Local preference indicates priority given to a particular route when more than one route exists to the same destination. A higher local preference indicates a more preferred route. Local preference is local to this autonomous system and is exchanged only with IBGP peers.

Syntax

set local-preference *value*

value	Preference value, ranging from 0 to 2147483647.

Syntax of the "no" Form

The *no* form of this command removes the local preference value:

no set local-preference *value*

Mode

Route-map configuration: **XSR(config-route-map)#**

Default

Preference value: 100.

Example

The following example configures the set local-preference value in the context of configuring a route map and match:

```
XSR(config)#route-map 1 permit 1
XSR(config-route-map)#match as-path 37
XSR(config-route-map)#set local-preference 400
```

Route map *1* uses the match as-path command that is referencing an as-path access list *37*. This list identifies the BGP routing updates to which the set local-preference command will apply. The relevant updates will have the value of their local preference set to *400*, which is higher than the default of *100*. Assuming that all of the BGP route selection criteria remain the same, the routes with the highest local preference will be chosen as the best routes to the identified destinations. This, however, applies only in multi-homed ASs as the local preference attribute impacts only which way the traffic leaves an AS if there are multiple exit points from it.

set metric

This command modifies the metric associated with routes that match a particular route map. This command can also be used to manipulate the value of the MED for matching BGP routes. Be sure that a match clause has been specified.

Metrics are values that the router uses to indicate preferred paths to networks. Updates with non-zero metrics are used for route selection inside the AS. BGP automatically compares metrics for routes to internal neighbors. You can use metric to select the best path when there are multiple alternatives. Routes with lower metric values are more preferred.

Syntax

set metric *metric-value*

metric-value	The value of the metric, ranging from 0 to 2,147,483,647.

Syntax of the "no" Form

The *no* form of this command removes the metric value:

no set metric *metric-value*

Mode

Route-map configuration: **`XSR(config-route-map)#`**

Default

The dynamically-learned metric value.

Example

The following example displays how the set metric command is used to update the value of the MED value for BGP routes that are advertised to an external neighbor:

```
XSR(config)#access-list 66 permit 10.0.0.0 255.0.0.0
XSR(config)#route-map 1 permit 1
XSR(config-route-map)#match ip address 66
XSR(config-route-map)#set metric 20
XSR(config-route-map)#exit
XSR(config)#route-map 1 permit 2
XSR(config-route-map)#set metric 30
XSR(config-route-map)#exit
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.1.1 route-map 1 out
```

The set metric command is used to change the value of the MED, which impacts the flow of inbound traffic into a multi-homed AS. All of the outbound updates leaving this router and matching ACL *66* will have MED value of *20* assigned to them. All of the remaining updates will have the MED value of *30*. A lower value of MED is preferred in the BGP route selection process.

set origin

This command assigns a value to the origin attribute in the BGP routing update message which impacts BGP route selection. Ensure that a match clause has been specified.

This attribute indicates where a routing update is derived. BGP prefers routes with the lowest origin type: IGP is preferred over EGP and EGP is preferred over incomplete.

Syntax

`set origin` {*`igp`* | *`egp`* | *`incomplete`*}

`igp`	Sets BGP origin code to Interior Gateway Protocol (IGP).
`egp`	Sets BGP origin code to Exterior Gateway Protocol (EGP).
`incomplete`	Sets BGP origin code to *unknown*.

Syntax of the “no” Form

The *no* form of this command removes BGP origin coding:

`no set origin` {*`igp`* | *`egp`* | *`incomplete`*}

Mode

Route-map configuration: **`Router(config-route-map)#`**

Default

The default value for this command is the default value for the origin code. The default value for the origin code is incomplete for routes that are advertised into BGP by means of the redistribute command.

Example

The following example configures the set origin value for redistributed static routes:

```
XSR(config)#route-map 1 permit 1
XSR(config-route-map)#set origin igp
XSR(config-route-map)#exit
XSR(config)#router bgp 100
XSR(config-router)#redistribute static route-map 1
```

set weight

This command specifies the weight value for matching BGP routing table entries. Be sure that a match clause has been specified.

Weight is used for best path selection and is assigned locally to the router. It is not propagated or carried through any route updates. Routes with a higher weight are preferred when multiple routes exist to the same destination.

Syntax

`set weight` *`weight`*

`weight`	Weight is local to the XSR on which it is configured, and it is not propagated in BGP routing update messages. But, it is the first value considered in the BGP route selection process. Routes with the higher weight are prefered over alternate routes to the same destinations but with a lower weight. Range: 0 to 65535.

Syntax of the “no” Form

The *no* form of this command removes the weight value:

`no set weight` *`weight`*

Mode

Route-map configuration: **`Router(config-route-map)#`**

Defaults

- Routes advertised into BGP via redistribution or the **`network`** command: 32768
- Routes advertised by a BGP neighbor: 0

Example

The following example configures the weight parameter in the context of configuring route map *1* and applying it to updates arriving from two remote neighbors:

```
XSR(config)#ip as-path access-list 67 permit "^101 .*"
XSR(config)#ip as-path access-list 57 permit "^102 .*"
XSR(config)#route-map 1 permit 1
XSR(config-route-map)#match as-path 67
XSR(config-route-map)#set weight 6000
XSR(config-route-map)#exit
XSR(config)#route-map 1 permit 2
XSR(config-route-map)#match as-path 57
XSR(config-route-map)#set weight 5000
XSR(config-route-map)#exit
XSR(config)#router bgp 100
XSR(config-router)#neighbor 192.168.1.1 remote-as 101
XSR(config-router)#neighbor 192.168.2.1 remote-as 102
XSR(config-router)#
XSR(config-router)#neighbor 192.168.1.1 route-map 1 in
XSR(config-router)#neighbor 192.168.2.1 route-map 1 in
```

The two instances of route map *1* perform a match on IP as-path access lists *67* and *57*, in that order with a weight of *6000* for updates matching ACL *67*, and *5000* for updates matching ACL *57*. If the same destinations are advertised by all two remote neighbors, the outbound traffic from this router will be directed to the neighbor who had a match on ACL *67*, as those routes will have the highest value of the weight parameter.

BGP Clear and Show Commands

clear ip bgp

This command resets one or more BGP connections, by either a hard or soft reset. Soft resets are preferred because they are less disruptive overall to internetworking. BGP connections must be reset whenever the BGP routing policy is changed by means of one of the following:

- BGP-related access lists
- BGP-related weights
- BGP-related distribution lists
- Specification of the BGP timer
- BGP administrative distance
- BGP-related route maps
- BGP neighbor configuration

Two options for *soft* reset are:

- Route refresh is supported depending on whether the route refresh capability has been negotiated during the OPEN session
- Stored updates (explicit **`neighbor soft-reconfiguration`**)

Syntax

clear ip bgp {* | *address* | **peer-group** *peer-group-name*} [**soft** [**in** | **out**]]}

*	A wild card which resets all current BGP sessions.
address	Resets the indicated BGP neighbor.
peer-group-name	Resets the indicated BGP peer group.
soft	Performs a soft reconfiguration.
in	Triggers an inbound soft reconfiguration.
out	Triggers an outbound soft reconfiguration.

Mode

Privileged EXEC: **XSR#**

Examples

This example displays all BGP connections and neighbors cleared by means of a hard reset, the most drastic way of clearing BGP links.

```
XSR#clear ip bgp *
```

The following example displays a soft inbound reset with neighbor *192.168.11.1*:

```
XSR#clear ip bgp 192.168.11.1 soft in
```

clear ip bgp dampening

This command resets BGP dampening parameters to the system default and unsuppresses suppressed routes.

Syntax

clear ip bgp {**dampening** [*ip-address mask*]}

ip-address	The network to clear damping information on.
mask	The network mask to clear damping information on.

Mode

Privileged EXEC: **XSR#**

Examples

The following example clears route dampening information about the route to all routers and unsuppresses suppressed routes:

```
XSR#clear ip bgp dampening
```

The following example clears route dampening information about the route to network *12.0.0.0* and unsuppresses its suppressed routes:

```
XSR# clear ip bgp 12.0.0.0 255.0.0.0
```

show ip bgp

This command displays entries in the BGP routing table.

Syntax

show ip bgp [*network*][*network-mask*][*longer-prefixes*]

network	Number of a network in the BGP routing table.
network-mask	All BGP routes matching the address and mask pair.
longer-prefixes	Routes and specific routers are displayed.

Mode

EXEC configuration: **XSR>**

Examples

The following is sample output from the command:

```
XSR#show ip bgp

Local router ID is 1.1.1.4
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? = incomplete

Network              Next Hop       Metric LocPrf Weight Path
*> 192.4.4.0/24         192.168.72.100      0    300    100 300 ?
*> 192.1.1.0/24         192.168.72.100      0    300    100 300 ?
*  55.5.5.0/24          52.52.52.3               200    100 200 ?
*> 55.5.5.0/24          192.168.72.100      0    300    100 300 ?
*> 6.6.6.2/32           192.168.72.100      0    300    100 300 ?

Local Router ID: IP Address of the router
Status codes:
```

- s – the bgp table entry is suppressed
- * - the bgp table entry is valid
- > - the bgp table entry is the best entry for the network
- i – the bgp table entry is learned via IBGP

```
Origin Codes:
```

- i – Entry originated from an IGP
- e – Entry originated from an EGP
- ? – Entry originated from an unknown source (i.e redistribution)

Display Parameters

Network	IP address of destination network.
Next Hop	IP address of the next hop to the destination network.
Metric	Value of Multi-Exit Descriminator.
LocPrf	Value of Local Preference.
Weight	Weight of the route.
Path	AS path to the destination network.

The following is sample output from the command:

```
XSR#show ip bgp 55.5.5.0/24

BGP routing table entry for 55.5.5.0 255.255.255.0
Paths: (2 available, learned over EBGP)
AS Path 200, Aggregator 500  1.2.3.4
Next Hop 52.52.52.3 from 52.52.52.3 (52.52.52.3)
Origin ?, localpref 200, weight 100, atomic, valid

BGP routing table entry for 55.5.5.0 255.255.255.0
Paths: (2 available, best #1, learned over EBGP)
AS Path 300
Next Hop 192.168.72.100 from 192.168.72.100 (192.168.72.100)
Origin ?, localpref 300, med 0, weight 100, valid, best
```

show ip bgp community

This command displays routes associated with BGP communities.

Syntax

show ip bgp community community-number | internet | local-AS | no-export | no-advertise

community-number	Community number, ranging from 1 to 4,294,967,295.
internet	Well-known Internet community.
local-AS	Well-known community specifying that routes with this value should not be sent outside a local AS.
no-export	Well-known community specifying that routes with this value should not be advertised outside a BGP confederation boundary.
no-advertise	Well-known community specifying that routes with this value should not be advertised to any other.

Mode

EXEC configuration: **XSR>**

Example

The following is sample output from the command:

```
XSR#show ip bgp community 400

Local router ID is 1.1.1.4
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? = incomplete

Network                    Next Hop       Metric LocPrf Weight Path
*> 192.4.4.0/24         192.168.72.100      0    100    100  300 ?
*> 192.1.1.0/24         192.168.72.100      0    100    100  300 ?
*> 66.6.6.2/32          192.168.72.100      0    100    100  300 ?
*> 55.5.5.0/24          192.168.72.100      0    100    100  300 ?
*> 6.6.6.2/32           192.168.72.100      0    100    100  300 ?
```

show ip bgp community-list

This command displays routes that are permitted by the indicated BGP community list.

Syntax

show ip bgp community-list {community-list-number | [exact-match]}

community-list-number	Community list number. Range: 1 to 199.
exact-match]	Routes displayed by exact match.

Mode

EXEC configuration: **XSR>**

Example

The following is sample output from the command:

```
XSR#show ip bgp community community-list 1

Local router ID is 1.1.1.4
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? = incomplete

Network                    Next Hop       Metric LocPrf Weight Path
*> 192.4.4.0/24         192.168.72.100      0    100    100 300 ?
*> 192.1.1.0/24         192.168.72.100      0    100    100 300 ?
*> 66.6.6.2/32          192.168.72.100      0    100    100 300 ?
*> 55.5.5.0/24          192.168.72.100      0    100    100 300 ?
*> 6.6.6.2/32           192.168.72.100      0    100    100 300 ?
```

show ip bgp dampened-paths

This command displays BGP routes suppressed due to dampening.

Syntax

```
show ip bgp dampened-paths
```

Mode

EXEC configuration: **XSR>**

Example

The following is sample output from the command:

```
XSR#show ip bgp dampened-paths
Local router ID is 1.1.1.4
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? = incomplete

*> 192.4.4.0/24         192.168.72.100      0    100    100 300 ?
*> 192.1.1.0/24         192.168.72.100      0    100    100 300 ?
```

show ip bgp filter-list

This command displays routes conforming to a specified filter list.

Syntax

```
show ip bgp filter-list access-list-number
```

access-list-number	Number of an AS path ACL. Range: 1 to 199.

Mode

EXEC configuration: **XSR>**

Example

The following example is sample output from the command:

```
XSR#show ip bgp filter-list 2
Local router ID is 1.1.1.4
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? = incomplete

Network                 Next Hop       Metric LocPrf Weight Path
*> 192.4.4.0/24         192.168.72.100      0    100    100 300 ?
*> 192.1.1.0/24         192.168.72.100      0    100    100 300 ?
*> 66.6.6.2/32          192.168.72.100      0    100    100 300 ?
*> 55.5.5.0/24          192.168.72.100      0    100    100 300 ?
*> 6.6.6.2/32           192.168.72.100      0    100    100 300 ?
```

show ip bgp inconsistent-as

This command displays routes that have incomplete originating ASs.

Syntax

show ip bgp inconsistent-as

Mode

EXEC configuration: **XSR>**

Example

The following is sample output from the command:

```
XSR#show ip bgp inconsistent-as
Local router ID is 1.1.1.4
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? = incomplete

Network                  Next Hop       Metric LocPrf Weight Path
*> 192.4.4.0/24         192.168.72.100      0    100    100 300 ?
*> 192.1.1.0/24         192.168.72.100      0    100    100 300 ?
*> 66.6.6.2/32          192.168.72.100      0    100    100 300 ?
*> 55.5.5.0/24          192.168.72.100      0    100    100 300 ?
*> 6.6.6.2/32           192.168.72.100      0    100    100 300 ?
```

show ip bgp neighbors

This command displays information about TCP and BGP connections to neighbors.

Syntax

show ip bgp neighbors [*neighbor-address*]

neighbor-address	The IP address of the neighbor whose routes the XSR has learned from. If omitted, all neighbors are displayed.

Mode

EXEC configuration: **XSR>**

Example

The following is sample output from the command. The output is filtered to show only that the 192.168.72.100 neighbor and the route refresh capability has been exchanged with this neighbor.

```
XSR#show ip bgp neighbors 192.168.72.100

BGP neighbor is 192.168.72.100 remote AS 300 external link
 BGP version 4, remote router ID 192.168.72.100
 BGP state = ESTABLISHED
```

```
Hold time is 90, keepalive interval is 30 seconds
Neighbor capabilities:
  Route Refresh: advertised & received
  Address family IPv4 Unicast: advertised & received
Received 11 messages, 1 notifications
Sent 10 messages, 1 notifications,  0 in queue
Route Refresh request: received 0 sent 0
Last reset: Peer connection reset
3 accepted prefixes
Outgoing update AS path filter list is 33
Route map for outgoing advertisements is 60
```

Display Parameters

BGP neighbor	IP address of the BGP neighbor and its AS number. If the neighbor is in the same AS as the router, then the link between them is internal (IBGP), otherwise it is considered external (EBGP).
BGP neighbor	AS of the neighbor.
external link	This is an EBGP peer.
BGP version	BGP version used to communicate with the peer.
remote router ID	IP address of the neighbor.
BGP state	Internal state of the BGP connection.
Hold Time	Maximum interval, in seconds, that can elapse between messages from the peer.
keepalive interval	Interval, in seconds, between sending keepalive packets.
Neighbor capabilities	BGP capabilities advertised and received from this neighbor.
Route Refresh	Status of the route refresh capability.
Address family IPv4 Unicast	IP Version 4 unicast-specific properties.
Received	Sum of BGP messages received from this peer, including keepalives.
notifications	Sum of error messages received from the peer.
Sent	Sum of BGP messages sent to this peer, including keepalives.
notifications	Sum of error messages sent from this XSR to the peer.
Route refresh request	Sum of route refresh requests sent and received from this neighbor.
Last Reset	Previous reset reason.
accepted prefixes	Number of prefixes accepted.

show ip bgp peer-group

This command displays information about the BGP peer group belonging to the router that this command is entered on.

Syntax

show ip bgp peer-group [*peer-group-name*][*summary*]

peer-group-name	Information about a specific peer group.
summary	Summary status of all peer group members.

Mode

EXEC configuration: **XSR>**

Example

The following is sample output from the command:

```
XSR#show ip bgp peer-group external

BGP peer group is external
 BGP version 4
 Minimum time between advertisement runs is 0 seconds
 peer-group is external, members
 18.1.1.3 192.168.72.19

XSR#show ip bgp peer-group external summary

Neighbor        V     AS  MsgRcvd MsgSent   InQ    OutQ  State
192.168.72.19   4    400      157     169     0       0  ESTAB
18.1.1.3        4    400      157     164     0       0  ESTAB
```

show ip bgp regexp

This command displays BGP AS paths that match the indicated regular expression.

Syntax

show ip bgp regexp regexp

regexp	The regular expression to match BGP AS paths.

Mode

EXEC configuration: **XSR>**

Example

The following is sample output from the command:

```
XSR#show ip bgp regexp 300$
```

```
Local router ID is 1.1.1.4
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? = incomplete

Network                    Next Hop       Metric LocPrf Weight Path
*> 192.4.4.0/24          192.168.72.100      0    100    100 300 ?
*> 192.1.1.0/24          192.168.72.100      0    100    100 300 ?
*> 66.6.6.2/32           192.168.72.100      0    100    100 300 ?
*> 55.5.5.0/24           192.168.72.100      0    100    100 300 ?
*> 6.6.6.2/32            192.168.72.100      0    100    100 300 ?
```

show ip bgp summary

This command displays status for all BGP connections.

Syntax

```
show ip bgp summary
```

Mode

EXEC configuration: **XSR>**

Example

The following is sample output from the command:

```
XSR#show ip bgp summary

Neighbor        V     AS  MsgRcvd MsgSent   InQ    OutQ  State
192.168.72.19   4    400      177     189     0       0  ESTAB
18.1.1.3        4    400      177     184     0       0  ESTAB
52.52.52.3      4    200      186     188     0       0  ESTAB
192.168.72.100  4    300      177     186     0       0  ESTAB
```

Display Parameters

Neighbor	IP address of the neighbor.
V	BGP version spoken to the neighbor.
AS	AS number.
MsgRcvd	BGP messages received from a neighbor.
MsgSent	BGP messages sent to a neighbor.
InQ	Number of messages from a neighbor is waiting to be processed.
OutQ	Number of messages waiting to be sent to a neighbor.
State	Current state of the BGP session.

show route-map

This command displays configured route maps and information about policy maps that are referenced.

Syntax

show route-map [*map-number*]

map-number	The number of a route map, ranging from 1 to 199.

Mode

EXEC configuration: **XSR>**

Example

The following is sample output from the command:

```
XSR#show route-map

route-map 1, permit, sequence 1
 Match clauses:
  community-list 1
 Set clauses:
  local-preference 300

route-map 1, permit, sequence 2
 Match clauses:
  community-list 2
 Set clauses:
  local-preference 200

route-map 2, permit
 Match clauses:
  ip address 1
 Set clauses:
  community 100:100
```

BGP Debug Commands

debug ip bgp

This command displays information related to processing of the BGP. Like all XSR debug commands, it is set to privilege level 15 by default.

Syntax

debug ip bgp [*events* | *updates*]

events	Displays BGP events.
updates	Displays BGP updates.

Syntax of the "no" Form

The *no* form of this command disables debugging output:

no debug ip bgp [**events** | **updates**]

Mode

EXEC configuration: **XSR>**

Default

BGP debugging is disabled.

Examples

The following is sample output with the *events* option chosen:

```
XSR#debug ip bgp events

BGP: Event:STOP, Nbr:192.168.2.1, AS:300, Skt:0, State:IDLE

BGP: Event:START, Nbr: 192.168.2.1, AS:300, Skt:0,
State:PEND_START

BGP: Event:START, Nbr: 192.168.2.1, AS:300, Skt:2, State:CONNECT

BGP: Event:TCP_OPEN, Nbr: 192.168.2.1, AS:300, Skt:2, State:OPENSENT

BGP: Event:RX_OPEN, Nbr: 192.168.2.1, AS:300, Skt:2, State:OPENCONFIRM

BGP: Event:RX_KEEP, Nbr: 192.168.2.1, AS:300, Skt:2, State:ESTABLISHED

BGP: Event:RX_UPDATE, Nbr: 192.168.2.1, AS:300, Skt:2, State:ESTABLISHED

BGP: Event:KEEP_EXP, Nbr: 192.168.2.1, AS:300, Skt:2, State:ESTABLISHED

BGP: Debug event generated from the BGP process
Event: BGP event that has been processed
Nbr: Neighbor IP address
AS: AS number
Skt: Socket identifier
State: State of the BGP connection
```

The following is sample output with the *updates* option chosen:

```
XSR#debug ip bgp updates
BGP: Rx Update. Nbr: 192.168.2.1, w/ attr: Origin:? AS_SEQ Path:300 Next
Hop:192.168.2.2 Med:0
BGP: Rx NLRI. Nbr: 192.168.2.1, Prefix:6.6.6.0, Len:24
BGP: Rx NLRI. Nbr: 192.168.2.1, Prefix:7.7.7.0, Len:24
BGP: Rx NLRI. Nbr: 192.168.2.1, Prefix:8.8.8.0, Len:24

BGP: Tx Update. Nbr: 192.168.2.1, w/ attr: Origin:? AS_SEQ Path:100 Next
Hop:192.168.2.2
BGP: Tx NLRI. Nbr: 192.168.2.1, Prefix:5.0.0.0, Len:8
BGP: Tx NLRI. Nbr: 192.168.2.1, Prefix:10.0.0.0, Len:8
BGP: Tx NLRI. Nbr: 192.168.2.1, Prefix:2.0.0.0, Len:8
```

Display Parameters

BGP	Debug event generated by the BGP process.
Rx Update	Update message has been received.
Tx Update	Update message being transmitted.
Nbr	Neighbor IP address.
w/ attr	Path Attributes in the update message.
Origin	Origin of the path.
AS_SEQ Path	AS Sequence Path list.
Next Hop	Next Hop IP address.
Med	Multi-exit discriminator.
Rx NLRI	Received Network Layer reachability information.
Prefix	Network IP address.
Len	Length of prefix mask.
Tx NLRI	Transmitted Network Layer reachability information.

show ip traffic

This command display BGP statistics among other IP data.

Syntax

```
show ip traffic
```

Mode

EXEC configuration: **XSR>**

Example

The following sample outputdisplays only BGP-specific data:

```
XSR#show ip traffic

BGP Statistics:
  Rcvd:   184 total
          3 opens, 0 notifications, 4 updates
          177 keepalives, 0 route-refresh
  Sent:   186 total
          4 opens, 0 notifications, 6 updates
          176 keepalives, 0 route-refresh
```

7

Configuring IP Multicast

Observing Syntax and Conventions

The CLI command syntax and conventions use the notation described below.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[x]	[ ] Square brackets indicate an optional parameter (*italic*)
[x | y | z]	[|] Square brackets with vertical bar indicate a choice of values
{x | y | z}	{ | } Braces with vertical bar indicate a choice of a required value
[x {y | z}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<*xx*>)	*xx* signifies the interface, class map, policy map or other value you specify; e.g., **F1**, **G3**, **S2/1.0**, **<Your Name>**. **F** indicates a FastEthernet, and **G** a GigabitEthernet interface.
Next Mode entries display the CLI prompt after a command is entered.	
Sub-commands are displayed in ***red*** text.	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

IGMP and Generic Multicast Commands

The following command sets define IP Multicast functionality on the XSR, including:

- "PIM Commands" on page 7-89.
- "IGMP Clear and Show Commands" on page 7-95.

ip multicast-routing

This command enables/disables multicast routing and multicast switching.

Syntax

```
ip multicast-routing
```

Syntax

The *no* form of the command disables the multicast service:

no ip multicast-routing

Mode

Global configuration: **XSR(config)#**

Default

Disabled

Example

In the following example, multicast service is *enabled* on the XSR:

```
XSR(config)#ip multicast-routing
```

ip igmp version

This command manually sets the IGMP version on a local interface.

Syntax

ip igmp version *version_number*

version_number	IGMP version number, ranging from 1 to 3.

Syntax of the “no” Form

The *no* form of this command sets the default value.

no ip igmp version

Mode

Interface configuration: **XSR(config-if<xx>)#**

Default

IGMP Version 2

Example

The following example sets the IGMP version number to *3*:

```
XSR(config)#interface FastEthernet 1
XSR(config-if<F1>)#ip igmp version 3
```

ip igmp join

This command manually joins a multicast group to a local interface.

Syntax

ip igmp join-group *group-address*

group-address	Address of the multicast group.

Syntax of the "no" Form

The *no* form of this command cancels membership in a group:

no ip igmp join-group *group-address*

Mode

Interface configuration: **XSR(config-if<xx>)#**

Example

The following example joins the XSR to multicast group *225.2.2.1*:

```
XSR(config-if<F1>)#ip igmp join-group 225.2.2.1
```

ip igmp last-member-query-count

This command configures the retransmit count at which the XSR sends IGMP group-specific host query messages.

Syntax

ip igmp last-member-query-count *count*

count	Retransmit count, ranging from 1 to 7.

Syntax of the "no" Form

The *no* form of this command sets this count to the default:

```
no ip igmp last-member-query-count
```

Mode

Interface configuration: **XSR(config-if<xx>)#**

Default

2

Example

The following example changes the IGMP group-specific host query retransmit count to *3*:

```
XSR(config-if<F1>)#ip igmp last-member-query-count 3
```

ip igmp last-member-query-interval

This command sets the frequency at which IGMP group-specific host query messages are sent.

Syntax

ip igmp last-member-query-interval *interval*

interval	Frequency to send IGMP group-specific host query messages, ranging from 100 to 65535 milliseconds.

Syntax of the "no" Form

The *no* form of this command sets this frequency to the default:

no ip igmp last-member-query-interval

Mode

Interface configuration: **XSR(config-if<xx>)#**

Default

1000 milliseconds

Example

This example changes the IGMP group-specific host query message interval to 2 seconds:

```
XSR(config-if<F1>)#ip igmp last-member-query-interval 2000
```

ip igmp query-interval

This command configures the frequency at which the XSR sends IGMP host query messages.

Syntax

ip igmp query-interval *seconds*

seconds	Frequency to send IGMP host query messages, ranging from 1 to 32767 seconds.

Syntax of the "no" Form

The *no* form of this command sets this frequency to the default value:

no ip igmp query-interval

Mode

Interface configuration: **XSR(config-if<xx>)#**

Default

125 seconds

Example

This example changes the frequency which IGMP host-query messages are sent to *3* minutes:

```
XSR(config-if<F1>)#ip igmp query-interval 180
```

ip igmp query-max-response-time

This command configures the maximum response time advertised in IGMP queries.

Syntax

ip igmp query-max-response-time *seconds*

seconds	Maximum response time advertised in IGMP queries.

Syntax of the “no” Form

The *no* form of this command sets this response time to the default:

```
no ip igmp query-max-response-time
```

Mode

Interface configuration: **XSR(config-if<xx>)#**

Default

10 seconds

Example

The following example sets a maximum response time of *8* seconds:

```
XSR(config-if<F1>)#ip igmp query-max-response-time 8
```

ip igmp querier-timeout

This command sets the timeout period before the XSR takes over as the querier for the interface after the previous querier has stopped querying.

Syntax

ip igmp querier-timeout *seconds*

seconds	Interval the XSR waits after the previous querier has stopped querying and before it takes over as the querier, ranging from 2 to 65535 seconds.

Syntax of the “no” Form

The *no* form of this command sets this response time to the default value:

```
no ip igmp querier-timeout
```

Mode

Interface configuration: **XSR(config-if<xx>)#**

Default

Two times the query interval

Example

The following example sets the XSR to wait *30* seconds from the time it received the last query before it takes over as the querier for the interface:

```
XSR(config-if<F1>)#ip igmp querier-timeout 30
```

ip multicast ttl-threshold

This command sets the Time-To-Live (TTL) threshold of packets being forwarded out an interface.

Syntax

ip multicast ttl-threshold *ttl-value*

ttl-value	Time-to-live value, ranging from 0 to 255 hops.

Syntax of the “no” Form

The *no* form of this command sets this threshold to the default value:

```
no ip multicast ttl-threshold
```

Mode

Interface configuration: **XSR(config-if<xx>)#**

Default

Zero - all multicast packets are forwarded out the interface.

Example

The following example sets the TTL threshold on a border router to *20*. Multicast packets must have a TTL greater than *20* in order to be forwarded out this interface:

```
XSR(config-if<F1>)#ip multicast ttl-threshold 20
```

PIM Commands

ip pim sparse-mode

This command enables Protocol Independent Multicast (PIM) Sparse Mode (SM) on a local interface.

Syntax

```
ip pim sparse-mode
```

Syntax of the “no” Form

The *no* form of this command disables PIM on an interface:

```
no ip pim sparse-mod
```

Mode

Interface configuration: **XSR(config-if<xx>)#**

Default

PIM-SM is disabled on an interface

Example

The following example enables PIM sparse mode on *F1*:

```
XSR(config-if<F1>)#ip pim sparse-mode
```

ip pim bsr-border

This command specifies an interface so BootStrap Router (BSR) messages are not sent or received through an interface.

Syntax

```
ip pim bsr-border
```

Syntax of the “no” Form

The *no* form of this command removes the BSR border setting:

```
no ip pim bsr-border
```

Mode

Interface configuration: **XSR(config-if<xx>)#**

Example

The following example sets interface *F1* as the PIM domain border:

```
XSR(config-if<F1>)#ip pim bsr-border
```

ip pim bsr-candidate

This command enables the XSR to announce its candidacy as a BootStrap Router (BSR).

Syntax

ip pim bsr-candidate *type number* [*hash-mask-length* [*priority*]]

type number	Interface from which the BSR address is derived, to make it a candidate. This interface must be enabled with PIM.
hash-mask-length	Length of a mask that is used to be ANDed with the group address before the hash function is called. All groups with the same seed hash (correspond) to the same Rendezvous Point (RP). This option provides one RP for multiple groups.
priority	Preference value, ranging from 0 to 255. The BSR with the larger priority is preferred. If priority values are the the same, the IP address breaks the tie. The BSR candidate with the higher IP address is preferred.

Syntax of the “no” Form

The *no* form of this command removes this XSR as a BSR candidate:

```
no ip pim bsr-candidate
```

Mode

Global configuration: **XSR(config)#**

Defaults

- BSR candidate is not enabled with this router.
- Priority: 0

Example

The following example configures the IP address of the router on *F1* to be a candidate:

```
XSR(config)#ip pim bsr-candidate FastEthernet 1
```

ip pim dr-priority

This command sets the priority for which a router is elected as the Designated Router (DR).

Syntax

ip pim dr-priority *priority-value*

priority-value	Preference value, ranging from 0 to 4294967294, to set the priority of the router for selection as the DR.

Syntax of the "no" Form

The *no* form of this command disables the DR functionality:

no ip pim dr-priority

Mode

Interface configuration: **XSR(config-if<xx>)#**

Defaults

- DR functionality is disabled on the interface
- DR-priority: 1

Example

The following example sets the DR priority value of *F1* to *20*:

```
XSR(config-if<F1>)#ip pim dr-priority 20
```

ip pim message-interval

This command configures the frequency at which a Protocol Independent Multicast Sparse Mode (PIM-SM) router sends periodic join and prune messages.

Syntax

ip pim message-interval *seconds*

seconds	Interval to send periodic PIM-SM join and prune messages. Range: 1 to 65535.

Syntax of the "no" Form

The *no* form of this command sets the interval to the default value:

no ip pim message-interval

Mode

Interface configuration: **XSR(config-if<xx>)#**

Default

60 seconds

Example

The following example changes the PIM-SM message interval to *120* seconds:

```
XSR(config-if<F1>)#ip pim message-interval 120
```

ip pim query-interval

This command sets the frequency of Protocol Independent Multicast (PIM) router query messages.

Syntax

ip pim query-interval *seconds*

seconds	Interval to send periodic PIM router query messages. Range: 1 to 65535.

Syntax of the "no" Form

The *no* form of this command sets the interval to the default value:

```
no ip pim query-interval
```

Mode

Interface configuration: **XSR(config-if<xx>)#**

Default

30 seconds

Example

This example resets the PIM router query message interval to *60* seconds:

```
XSR(config-if<F1>)#ip pim query-interval 60
```

ip pim rp-address

This command sets the static Rendezvous Point (RP) for the specific multicast group. (Dynamically learned RP always has a higher priority than statically configured RP.)

Syntax

ip pim rp-address *rp-address* [*access-list*]

rp-address	IP address of a router to be a PIM RP. This is a unicast IP address in four-part, dotted notation.
access-list	ACL number defines for which multicast groups the RP should be used.

Syntax of the "no" Form

The *no* form of this command removes the static RP configuration:

no ip pim rp-address *rp-address*

Mode

Global configuration: **XSR(config)#**

Example

This example configures the RP used by the multicast groups within the range *225.1.1.0/24*:

```
XSR(config)#access-list 2 permit 225.1.1.0 0.0.0.255
XSR(config)#ip pim rp-address 192.168.2.5
```

ip pim rp-candidate

This command sets the XSR to advertise itself as a PIM candidate Rendezvous Point (RP) to the BSR. Only one candidate RP can be configured per box.

Syntax

ip pim rp-candidate *type number* [*group-list access-list*][*priority priority-value*]

type number	Interface whose IP address is advertised as a candidate RP address.
access-list	Standard IP access list number that defines the group prefixes that are advertised in association with the RP address.
priority	The priority of this candidate RP.
priority-value	Priority value, ranging from 0 to 255.

Syntax of the "no" Form

The *no* form of this command removes this XSR as an RP candidate:

no ip pim rp-candidate

Mode

Global configuration: **XSR(config)#**

Defaults

- The XSR is not configured as an RP candidate.
- DR priority is 192 by default if it becomes one.

Example

This example sets the XSR to advertise itself as a candidate RP to the BSR in its PIM domain:

```
XSR(config)#interface FastEthernet 1
XSR(config)#ip pim rp-candidate FastEthernet 1
```

ip pim regcksum wholepacket

This command changes the register checksum calculation to the industry standard.

Syntax

ip pim RegCksum wholepacket

Syntax of the “no” Form

The *no* command removes the static RP configuration:

no ip pim RegCksum wholepacket

Mode

Global configuration: **XSR(config)#**

Default

Checksum based on header only.

Example

The following example changes the calculation of the register packet to the industry standard:

```
XSR(config)#ip pim RegCksum wholepacket
```

ip pim spt-threshold

This command configures the threshold over which a PIM leaf router should join the shortest path source tree for the specified group.

Syntax

ip pim spt-threshold {*kbps*|**infinity**} [**group-list** *access-list*]

kbps	Traffic rate in kbps.
infinity	Never join the shortest path tree.
group-list *access-list*	Groups the threshold applies to. The value 0 applies the threshold to all groups.

Syntax of the “no” Form

The *no* form of this command restores the threshold to the default:

no ip pim spt-threshold

Mode

Global configuration: **XSR(config)#**

Default

The threshold is 0

Example

The following example sets the source tree switching threshold to 4 kbps:

```
XSR(config)#ip pim spt-threshold 4
```

IGMP Clear and Show Commands

clear ip mroute

This command deletes entries from the multicast table.

Syntax

clear ip mroute [*group-address*][*source-address*]

group-address	IP address of the multicast group.
source-address	IP address of the multicast source.

Mode

EXEC configuration: **XSR>**

show ip igmp groups

This command displays the multicast groups with receivers that are directly connected to the XSR and were learned through the Internet Group Management Protocol (IGMP).

Syntax

show ip igmp groups [*group-address* | *type number* | *summary*]

group-address	Address of the multicast group.
type	Interface type.
number	Interface number.
summary	A one-line, abbreviated summary of each entry in the IGMP groups table.

Mode

EXEC configuration: **XSR>**

Example

The following example displays sample responses:

```
XSR>show ip igmp groups
  Interface name:                FastEthernet1
```

```
State:                          Dynamic
Mode:                           Include
Current version:                V3
Group IP:                       232.1.1.1
Reporter IP:                    3.3.3.199
V1MEM exist timer:              0
V2MEM exist timer:              0
Member expire timer:            256
Source IP:                      6.6.6.10 (Forward state: YES, Timer:260)
```

Parameters in the Response

Group IP	Multicast group address.
Interface name	The interface through which the group membership is learned.
State	Dynamic learning or static configure.
Mode	Exclude or Include.
Reporter IP	Last host to report being a member of the multicast group.
V1MEM exist timer	V1 member existing timer.
V2MEM exist timer	V2 member existing timer.
Member expire timer	Group member expire timer.
Source IP	Sender IP address.
Forward state	Forward state for this source IP.
Timer	Source timer for this source IP.

show ip igmp interface

This command displays multicast-related information about an interface.

Syntax

show ip igmp interface [*type number*]

type	Interface type.
number	Interface number.

Mode

EXEC configuration: **XSR>**

Example

The following example displays sample responses:

```
XSRinterface
  Interface name:               FastEthernet2
  Interface state:              Up
  IGMP version:                 2
  Protocol owner:               PIM-SM
```

```
IGMP state:                     Enabled
Multicast ttl threshold: 0
Current query Interval:         125
Last Member Interval:           1
Querier timeout:                255
Max Response Timeout:         10
Current robust value:           2
Querier IP:                     1.1.1.2 (Self)
Query sending timer:            124
Group configured:               None
-------------------------------------------------------
Interface name:                 FastEthernet1
Interface state:                Up
IGMP version:                   3
Protocol owner:                 PIM-SM
IGMP state:                     Enabled
Multicast ttl threshold: 0
Current query Interval:         125
Last Member Interval:           1
Querier timeout:                255
Max Response Timeout:         10
Current robust value:           2
Querier IP:                     3.3.3.1 (Self)
Query sending timer:            124
Group configured:               225.1.1.1
-------------------------------------------------------
```

Parameters in the Response

Interface name	Interface type, number.
Interface state	Interface status.
IGMP version	IGMP version on this interface.
Protocol owner	Multicast routing protocol configured on this interface.
IGMP state	IGMP enable state.
Multicast ttl threshold	Multicast TTL threshold on this interface.
Configured query interval	Configured query interval on this interface.
Current query interval	Current query interval on this interface.
Last member interval	Last member interval on this interface.
Querier timeout	Querier timeout configured on this interface.
Max response timeout	Max response timeout configured on this interface.
Current robust value	Robust value on this interface.
Querier IP	Querier IP address.
Query sending timer	Query sending timer on this interface.
Group configured	Static groups configured on this interface.

show ip mroute

This command displays entries in the IP multicast routing table.

Syntax

show ip mroute [][*source-address*][*summary*]

group-address	IP address of the multicast group.
source-address	IP address of the multicast source.
summary	A one-line, abbreviated summary of each entry in the IP multicast routing table.

Mode

EXEC configuration: **XSR>**

Example

The following example displays sample responses:

```
XSR>show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, P - Pruned
       F - Register flag, T - SPT-bit set
Timers: Uptime/Expires
Interface state: Interface, Next-Hop, State/Mode

(*, 224.0.255.3), 5:29:15/00:01:14, RP is 192.168.26.2, flags:
  Incoming interface: FastEthernet1, RPF neighbor 10.3.35.1
  Outgoing interface list:
    FastEthernet0, Forward/Sparse, 5:29:15/0:01:57

(192.168.27.0/24, 224.0.255.3), 6:29:15/00:02:47, flags: TS
  Incoming interface: FastEthernet1, RPF neighbor 10.3.35.1
  Outgoing interface list:
    FastEthernet0, Forward/Sparse, 8:29:15/0:02:47
```

Parameters in the Response

Flags	Provides information about following entries: • *D - Dense:*- Entry is operating in dense mode. • *S - Sparse*: Entry is operating in sparse mode. • *C - Connected:* A member of the multicast group is present on the directly connected interface. • *P - Pruned:* Route has been pruned. • *F - Register flag:* Indicates that the software is Registering for a multicast source. • *T - SPT-bit set*: Indicates that packets have been received on the shortest path source tree.
(198.92.37.100/32, 224.0.255.1)	Entry in the IP multicast routing table. The entry consists of the IP address of the source router followed by the IP address of the multicast group. An asterisk (*) in place of the source router indicates all sources.
uptime	The interval in hours, minutes, and seconds the entry has been in the IP multicast routing table.
RP	Address of the rendezvous point (RP) router. For routers and access servers operating in sparse mode, this address is always 0.0.0.0.
flags	Information about the entry.
Incoming interface	Expected interface for a multicast packet from the source. If the packet is not received on this interface, it is discarded.
RPF neighbor	IP address of the upstream router to the source. *Tunneling* indicates that this router is sending data to the RP encapsulated in Register packets. The hexadecimal number in parentheses indicates to which RP it is registering. Each bit indicates a different RP if multiple RPs per group are used.
Outgoing interface list	Interfaces through which packets will be forwarded.
FastEthernet1	Name and number of the outgoing interface.
Forward/Sparse	Sparse-mode interface is in forward mode.
time/time (uptime/ expiration time)	Per interface, the interval in hours, minutes, and seconds the entry has been in the IP multicast routing table. Following the slash (/), the interval in hours, minutes, and seconds until the entry will be removed from the table.

show ip pim bsr

This command displays Bootstrap Router (BSR) version 2 information.

Syntax

```
show ip pim bsr
```

Mode

EXEC configuration: **XSR>**

Example

The following example displays sample responses:

```
XSR>#show ip pim bsr
PIMv2 Bootstrap information
This system is the Elected Bootstrap Router (BSR)
  BSR address: 192.168.27.1
  Uptime: 04:37:46, BSR Priority: 4, Hash mask length: 30
  Next bootstrap message in 00:00:03 seconds
This system is the Candidate Bootstrap Router (CBSR)
  Candidate BSR Address: 50.0.0.30 Priority: 0, Hash Mask Length: 30
```

Parameters in the Response

BSR address	IP address of the bootstrap router.
Uptime	Interval that this XSR has been up, in hours:minutes:seconds.
BSR Priority	Priority as set by the **ip pim bsr-candidate** command.
Hash mask length	Length of a mask (32 bits maximum) that is to be ANDed with the group address before the hash function is called. This value is configured by the **ip pim bsr-candidate** command.
Next bootstrap message in	Period (in hours:minutes:seconds) in which the next bootstrap message is due from this BSR.

show ip pim interface

This command displays data about interfaces set for Protocol Independent Multicast (PIM).

Syntax

show ip pim interface [*type number*]

type	Interface type.
number	Interface number.

Mode

EXEC configuration: **XSR>**

Example

The following example display sample responses:

```
XSR>show ip pim interface
PIM Interface Table
Address          Interface          Nbr Count     Hello Intvl      DR
30.0.0.20        FastEthernet1      0             30               30.0.0.20
40.0.0.20        FastEthernet2      2             30               40.0.0.40
```

Parameter Descriptions

Address	IP address of the next-hop router.
Interface	Interface type and number that is configured to run PIM.
Nbr Count	Number of PIM neighbors discovered through this interface.
Hello Intvl	The interval between Hello messages. The default is 30 seconds.
DR	IP address of the designated router on the LAN.

show ip pim neighbor

This command displays discovered Protocol Independent Multicast (PIM) neighbors.

Syntax

show ip pim neighbor [*type number*]

type	Interface type.
number	Interface number.

Mode

EXEC configuration: **XSR>**

Example

The following example shows sample responses:

```
XSR>#show ip pim neighbor
PIM Neighbor Table
Neighbor Address Interface DR Priority   Uptime    Expires   Mode
192.168.26.2     Ethernet0               15:38:16  0:01:25   Sparse
192.168.26.33    Ethernet0               13:33:20  0:01:05   Sparse (DR)
192.168.27.1     Ethernet1               15:33:20  0:01:08   Sparse (DR)
192.192.27.13    Ethernet1               16:56:06  0:01:04   Sparse
```

Parameters Descriptions

Neighbor Address	IP address of the PIM neighbor.
Interface	Interface type and number on which the neighbor is reachable.
DR Priority	The DR priority of the neighbor.
Uptime	Interval in hours, minutes, and seconds the entry has been in the PIM neighbor table.
Expires	Interval in hours, minutes, and seconds until the entry will be removed from the IP multicast routing table.
Mode	Mode in which the interface is operating.
(DR)	Indicates that this neighbor is a designated router on the LAN.

show ip pim rp

This command displays the active rendezvous points (RPs) that are cached with associated multicast routing entries.

Syntax

show ip pim rp [*group-address* | *mapping*]

group-address	Address of the group about which to display RPs.
mapping	Displays all group-to-RP mappings of which the XSR is aware.

Mode

EXEC configuration: **XSR>**

Example

The following example display sample responses:

```
XSR>show ip pim rp
Group: 224.2.240.20, RP: 192.168.10.13
Group: 224.1.127.155, RP: 192.168.10.13
Group: 224.2.127.154, RP: 192.168.10.13
Group: 224.2.128.153, RP: 192.168.10.13

XSR>show ip pim rp mapping
Group Address: 224.0.0.0 Mask: 240.0.0.0
    RP Address: 30.0.0.20 Holdtime: 150 Priority: 192
    RP Address: 50.0.0.40 Holdtime: 150 Priority: 192
```

Parameter Descriptions

Group	Address of the multicast group about which to display RP data.
RP	Address of the RP for that group.
Holdtime	The interval before the candidate RP expires.
Priority	The priority value for the candidate RP.

show ip pim rp-hash

This command displays the rendezvous point (RP) that is being selected for a specified group.

Syntax

show ip pim rp-hash {*group-address*}

group-address	Address of the group about which to display RPs.

Mode

EXEC configuration: **XSR>**

Example

The following example displays sample responses:

```
XSR>show ip pim rp-hash 239.1.1.1
             RP 192.168.27.12
```

Parameter Descriptions

RP	Address of the RP for the group specified (239.1.1.1).

8

Configuring the Point-to-Point Protocol

Observing Syntax and Conventions

The CLI Syntax and conventions use the notation described in the following table.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[*x*]	[ ] Square brackets indicate an optional parameter (*italic*)
[*x* | *y* | *z*]	[|] Square brackets with vertical bar indicate a choice of values
{*x* | *y* | *z*}	{ | } Braces with vertical bar indicate a choice of a required value
[*x* {*y* | *z*}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<*xx*>)	*xx* signifies interface type and number, e.g.: `F1`, `S2/1.0`, `D1`, `M57`, `G3. F` indicates a FastEthernet, and `G` a GigabitEthernet interface.
Next Mode entries display the CLI prompt after a command is entered.	
Sub-command headings are displayed in ***red*** text.	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

PPP Commands

This chapter defines Point-to-Point Protocol (PPP) service profiles, specify and monitor serial ports, and define Multilink PPP and Bandwidth Allocation Protocol (BAP) functionality in the following command sets:

- "PPP Debug, Clear and Show Commands" on page 8-97.
- "Multilink PPP Commands" on page 8-108.
- "Multilink Show Commands" on page 8-122.

encapsulation ppp

This command sets the Point-to-Point Protocol (PPP) as the encapsulation method used by a serial port. To use PPP encapsulation, the XSR must be configured with an IP routing protocol.

Note: If encapsulation is changed from one type to another, all related values of the current encapsulation and any sub-interface settings are deleted. Also, once encapsulation is set on an interface, any sub-interface of that port created later is automatically encapsulated. Finally, you must first enter the **`no encapsulation`** command to change the encapsulation type.

Syntax

```
encapsulation ppp
```

Syntax of the "no" Form

```
no encapsulation ppp
```

Default

No encapsulation

Mode

Interface configuration: **`XSR(config-if<xx>)`**

Example

The following example enables PPP encapsulation on Serial interface *1/0*:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation ppp
```

interface

This command selects a physical or virtual port for configuration as a router interface. The XSR supports ATM, BRI, Dialer, Fast/GigabitEthernet, Loopback, Multilink, Serial, or VPN interfaces. For configuration purposes, all serial ports and T1/E1/ISDN-PRI channel groups are treated as a serial interface.

Optionally, you can set up the Console port on the XSR 1800 series as a *WAN* interface for dial backup purposes (refer to the *Caution* below). Do so by entering **`0`** only.

Caution: Be aware that when you enable the Console port as a WAN port, you can no longer directly connect to it because it is in data communication mode. Your only access to the CLI will be to Telnet to an IP address of a configured port. Also, if your *startup-config* file does not configure any ports properly and sets up the console port as a serial interface, you will no longer be able to login and will have to press the Default button to erase your configuration. For details about configuring the Console with a modem, see "Chapter 2: Managing the XSR" in the *XSR User's Guide*.

Syntax

interface type *slot_num card_num port_num sub-interface_num*

type	ATM, BRI, Dialer, Fast/GigabitEthernet, Loopback, Multilink, Serial or VPN port.
slot_num	The NIM number ranging from *0* to *6* depending on the XSR model.
card_num	The NIM card number ranging from *1* to *2* depending on the NIM installed in the slot.
port_num	The physical port number ranging from: *0* (ATM), *0* to *1* (BRI), *0* to *255* (Dialer & VPN), *0* to *15* (Loopback), *1* to *32767* (Multilink), *0* to *3* (Serial), *1* to *2* (FastEthernet), *1* to *3* (GigabitEthernet), and *0* (Console). If a Serial port resides on a T1/E1 port, then channel group data must be added at the end of the string to mark which channel group of the T1/E1 port will be set: *card_num/NIM_num/ port_within_NIM*: *[channel-group_num].* For example, *0/2/1:15* sets channel-group 15 of the T1 or E1 port 1 in NIM slot 2 on the motherboard.
sub-interface_num	Number ranging from 1 to 30 (ATM, BRI & Serial), and 1 to 64 (Fast/GigabitEthernet).

Slots, cards, ports, and sub-interfaces are expressed as follows on the CLI:

0	The console port. (Only on the XSR 1800 series)
<0-0>/<1-2>/<0-3>	Slot, card, and port number.
<1-2>/<0-3>	Card and port number.
<1-2>/<0-3>.<1-30>	Card, port and sub-interface number.
<1-2>/<0-3>:<0-31>	Card, port and channel number.
<1-2>/<0-3>:<0-31> .<1-30>	Card, port, channel and sub-interface number.

Note: Leading zeros defined in *interface_num* can be omitted. For example, 0/1/2 is equivalent to 1/2.

Syntax of the “no” Form

The *no* command deletes the interface:

no interface serial port_num interface_num

Note: You cannot directly delete a Serial interface assigned to a T1/E1 channel group. You must instead delete a channel group to delete the Serial port.

Mode

Global configuration: **XSR(config)#**

Examples

This example selects interface serial *1/0* and sets PPP encapsulation:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation ppp
XSR(config-if<S1/0>)#no shutdown
```

The following example selects channel group *12* of the T1/E1 port1 on the second NIM card so that later configurations will apply to this serial port:

```
XSR(config)#interface serial 2/1:12
XSR(config-if<s2/1:12)#encapsulation ppp
XSR(config-if<S1/0>)#no shutdown
```

ppp authentication

This command specifies the type and order in which CHAP, MS-CHAP or PAP protocols are requested on the interface. Once CHAP, PAP authentication or both have been enabled, the XSR requires the remote device to prove its identity before allowing data traffic to flow.

PAP authentication requires the remote device to send a name and password to be checked against a matching entry in the local username database.

CHAP authentication sends a challenge to the remote device. The remote device must encrypt the challenge value with a shared secret and return the encrypted value and its name to the XSR in a response message. The XSR uses the remote device's name to look up the appropriate secret in the local username database. It uses the looked-up secret to encrypt the original challenge and verify that encrypted values match.

MS-CHAP is closely derived from the PPP CHAP with the exception that it uses MD4 as the hashing algorithm.

You may enable PAP or CHAP, MS-CHAP or all of them, in either order. If both methods are enabled, then the first method specified will be requested during link negotiation. If the peer suggests using the second method or simply refuses the first, then the second method is tried. Some remote devices support CHAP only and some PAP only. The order in which you specify the methods will be based on your concerns about the remote device's ability to correctly negotiate the appropriate method as well as your concern about data line security. PAP usernames and passwords are sent as *clear-text* strings and can be intercepted and reused. CHAP has eliminated most of the known security holes.

Enabling or disabling PPP authentication does not affect the XSR's willingness to authenticate itself to the remote device.

Note: If you specify CHAP authentication on one side of a connection, you should set CHAP on the other side as well.

Syntax

ppp authentication {any mix of **pap chap ms-chap**}

Possible parameter combinations include:

chap	Enables CHAP on a serial interface.
pap	Enables PAP on a serial interface.
ms-chap	Enables MS-CHAP on a serial interface.
chap pap	Preference of CHAP authentication before PAP.
pap chap	Preference of PAP authentication before CHAP.

ms-chap pap chap	Preference of MS-CHAP authentication, then PAP authentication, then CHAP.

Syntax of the "no" Form

The *no* form of this command disable PPP authentication:

```
no ppp authentication
```

Default

Not enabled

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example 1

Figure 8-1 shows two routers, Site A and Site B, attempting to authenticate each other using CHAP. The configuration example follows.

Figure 8-1 Authentication Configured on Both Peers

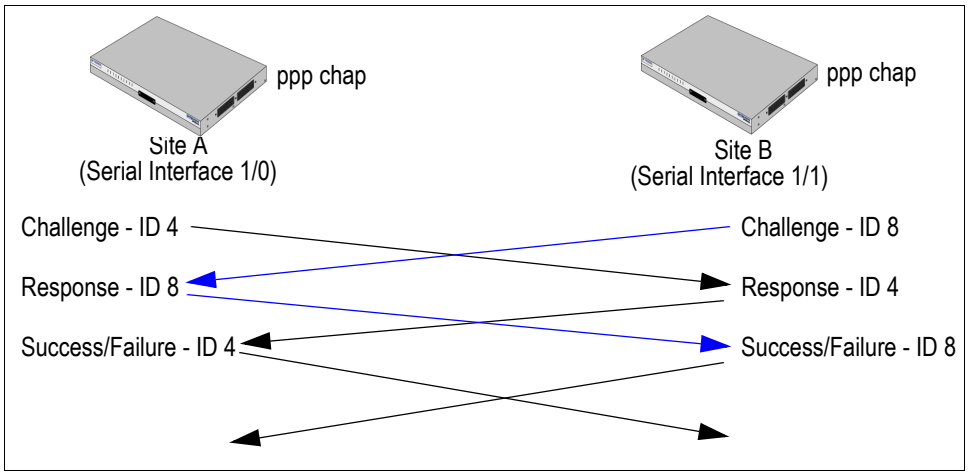


Figure 8-1 shows both routers send challenges and responses and either a failure or success. The following sample configuration illustrates the preceding example. On *Site A*, enter the following commands:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation ppp
XSR(config-if<S1/0>)#no shutdown
XSR(config-if<S1/0>)#ppp authentication chap
```

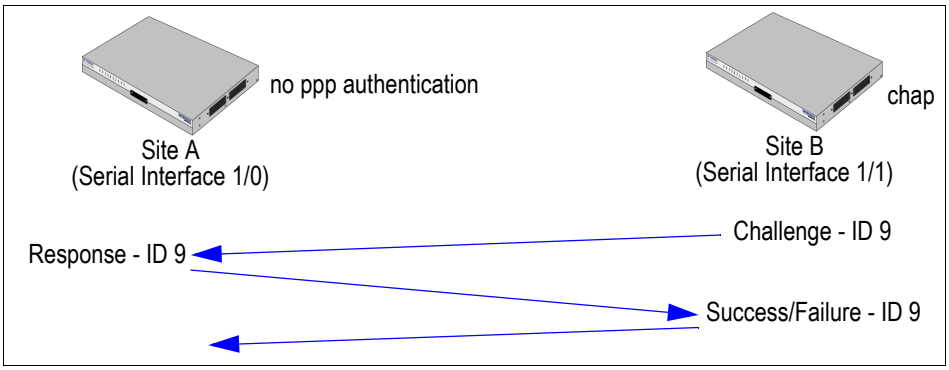
On *Site B*, enter the following commands:

```
XSR(config)#interface serial 1/1
XSR(config-if<S1/1>)#encapsulation ppp
XSR(config-if<S1/1>)#no shutdown
XSR(config-if<S1/1>)#ppp authentication chap
```

Example 2

Figure 8-2 shows two routers, Site A and Site B, and only one peer configured to do authentication (using chap) with only Site B issuing the challenge. The configuration example follows.

Figure 8-2 Authentication Configured on One Peer



Refer to the following sample configuration for the preceding example. On *Site A* enter the following commands:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation ppp
XSR(config-if<S1/0>)#ppp authentication chap
```

On *Site B* enter the following commands:

```
XSR(config)#interface serial 1/1
XSR(config-if<S1/1>)#encapsulation ppp
XSR(config-if<S1/1)#no ppp authentication
```

ppp chap

This command specifies a unique hostname on an interface, refuses CHAP authentication requests from peers, or uses a default password during CHAP authentication when no other password is available. It can enable multiple routers to appear to have the same hostname when using CHAP authentication.

This command can be used to set a default password during authentication challenges when the challenger's username cannot be found in the username list. It is also required when the challenger does not specify its name in the challenge packet and a default password must be sent. Be aware that this password is only used in response to challenges and is not used to authenticate the peer.

Syntax

ppp chap {**hostname** *hostname* | **refuse** | **password** *word*}

hostname	Alternate name sent in the CHAP challenge.
refuse	Refuse to authenticate using CHAP.
word	Default password sent to CHAP challenges when no passwords are available.

Syntax of the "no" Form

The *no* form of this command disables either function:

no ppp chap {**hostname** | **refuse** | **password**}

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Examples

The following example creates the alternate CHAP hostname *freud* and the default chap password *sigmund*:

```
XSR(config)#interface dialer 1
XSR(config-if<D1>)#encapsulation ppp
XSR(config-if<D1>)#ppp chap hostname freud
XSR(config-if<D1>)#ppp chap password sigmund
```

The following example enables CHAP authentication refusal:

```
XSR(config)#interface dialer 1
XSR(config-if<D1>)#encapsulation ppp
XSR(config-if<D1>)#ppp chap refuse
```

ppp keepalive

This command sets the keepalive timer on a Point-to-Point port. PPP keepalives are sent out as echo requests over the PPP port at specified intervals. They apply to any serial port on which PPP encapsulation is enabled. If you do not specify the interval the default interval is used.

When Link Quality Management (LQM) is enabled on the interface along with *ppp keepalive,* echo requests are disabled. Upon disabling the LQM feature echo requests will start again if *ppp keepalive* is still configured.

Syntax

ppp keepalive [*period*]

period	Keepalive period in seconds.

Syntax of the "no" Form

Use the *no* form of the command to disable the keepalives:

no ppp keepalive

Default

Enabled at 30 seconds

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example sets Serial interface *1/0* to have keepalive configured at *8*-second intervals:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation ppp
XSR(config-if<S1/0>)#no shutdown
XSR(config-if<S1/0>)#ppp keepalive 8
```

ppp lcp max-configure

This command configures the restart timer counter for the peak number of Configure-Requests sent out on a Point-to-Point interface. Using the Link Control Protocol (LCP), the command applies to any Serial, or Dialer port, or Fast/GigabitEthernet sub-interface on which PPP encapsulation is set. This counter totals the peak number of configure requests sent without receiving a Configure-Ack, Configure-Nak or Configure-Reject.

Syntax

ppp lcp max-configure *number*

number	Setting for the configure-request counter, ranging from 1 to 255.

Syntax of the “no” Form

The *no* command resets the counter to the default value:

```
no ppp lcp max-configure
```

Default

10

Mode

Serial, Dialer or Fast/GigabitEthernet sub-interface configuration: **XSR(config-if<xx>)#**

Example

The following example sets the LCP max-configure value at *2* requests:

```
XSR(config)#interface dialer 2
XSR(config-if<D2>)#ppp lcp max-configure 2
```

ppp lcp max-failure

This command configures the counter for the maximum number of Configure-Nak packets sent out on a Point-to-Point interface. Using the Link Control Protocol (LCP), the command applies to any Serial or Dialer port, or Fast/GigabitEthernet sub-interface on which PPP encapsulation is set. This counter totals the peak number of Configure-Nak packets to send; subsequent Nak packets are converted to Configure-Reject packets.

Syntax

ppp lcp max-failure *number*

number	Setting for the max-failure counter. Range: 1 to 255.

Syntax of the "no" Form

The *no* command resets the counter to the default value:

no ppp lcp max-failure

Default

5

Mode

Serial, Dialer or Fast/GigabitEthernet Sub-interface configuration: **XSR(config-if<xx>)#**

Examples

The following example sets the *lcp max-failure* value at *100* packets on Serial interface *2/1*:

```
XSR(config)#interface serial 2/1
XSR(config-if<S2/1>)#ppp lcp max-failure 100
```

The following example sets the lcp max-failure value at *200* packets on FastEthernet sub-interface *2/1.1*:

```
XSR(config)#interface fastethernet 2.1
XSR(config-if<F2/1:1>)#ppp lcp max-failure 200
```

ppp lcp max-terminate

This command configures the restart timer counter for the number of Terminate-Requests sent out on a Point-to-Point interface. Using the Link Control Protocol (LCP), the command applies to any Serial or Dialer port, or Fast/GigabitEthernet sub-interface on which PPP encapsulation is set.

This counter totals the peak number of terminate requests sent without receiving a Terminate-Ack before assuming that the peer cannot respond.

Syntax

ppp lcp max-terminate *number*

number	Setting for the terminate-request counter. Range: 1 to 255.

Syntax of the "no" Form

The *no* command resets the counter to the default value:

no ppp lcp max-terminate

Default

2

Mode

Serial, Dialer and Fast/GigabitEthernet Sub-interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example sets the terminate-request counter at *10* requests on Dialer interface *57*:

```
XSR(config)#interface dialer 57
XSR(config-if<D57>)#ppp lcp max-terminate 10
```

ppp max-bad-auth

This command permits multiple authentication failures. It configures a Point-to-Point interface not to reset itself immediately after an authentication failure but to allow a specified number of authentication retries. This command applies to any serial interface on which PPP encapsulation is enabled.

Syntax

ppp max-bad-auth *number*

number	Number of retries after which the interface resets itself.

Syntax of the "no" Form

Use the *no* form of this command to reset to the default (immediate reset):

```
no ppp max-bad-auth
```

Default

0

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example sets serial interface *1/0* to allow five additional retries after an initial authentication failure (for a total of *six* failed authentication attempts):

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation ppp
XSR(config-if<S1/0>)#no shutdown
XSR(config-if<S1/0>)#ppp authentication chap
XSR(config-if<S1/0>)#ppp max-bad-auth 6
```

ppp pap sent-username

This command configures a PAP username and clear text password for the specified interface. The value is used in the PAP authentication request packet to the peer.

Syntax

ppp pap sent-username [*username*] **password** [*password*]

username	Username sent in the PAP authentication request packet.
password	The clear text password sent in the PAP authentication request packet. Limit: up to 255 ASCII characters. Enclose password in double quotes if entering a string with spaces

Syntax of the "no" Form

Use the *no* form of this command to delete the username and password:

no pap sent-username

Default

No username or password

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

This example configuration of a the PAP authentication username of *jim* and a clear text PAP password of *evans* on serial interface *2/1*:

```
XSR(config)#interface serial 2/1
XSR(config-if<S2/1>)#encapsulation ppp
XSR(config-if<S1/1>)#no shutdown
XSR(config-if<S2/1>)#ppp pap sent-username jim pass evans
```

ppp peer default ip address

This command specifies the default IP address of a remote peer for use during PPP/IPCP negotiation if the peer requests it. The address is used when the remote peer sends a 0.0.0.0 IP address in the CONFIG REQUEST and asks the local system to assign an IP address. The address will not be used if the peer already has been assigned an IP address with its own local configuration.

This command can be used for Interface Serial, T1/E1 channel groups, BRI leased line with PPP encapsulated; Ethernet sub-interface and ATM sub-interface with PPPoE or PPPoA encapsulated. When used at the dialer interface, it applies to the Point-to-Point (P2P) dialer interface only. For Dialer Multipoint-to-Point interfaces, the **dialer map ip** command supplies the remote address associated with particular dialing numbers.

Note: The peer default IP address takes effect only when the peer is configured as *IP address negotiated*.

Syntax

ppp peer default ip address {*ip address*}

ip address	IP address of the remote peer.

Syntax of the "no" Form

Use the *no* form of this command to remove the IP address:

no ppp peer default ip address

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Examples

This example sets the peer's IP address on Serial interface *1/0*:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation ppp
XSR(config-if<S1/0>)#ppp peer default ip address 192.168.1.3
```

This example sets the peer's IP address on P2P Dialer interface 1:

```
XSR(config)#interface dialer 1
XSR(config-if<D1>)#encapsulation ppp
XSR(config-if<D1>)#ppp peer default ip address 10.10.10.1
```

This example sets the peer's IP address on M2P Dialer interface *2:*

```
XSR(config)#interface dialer 2 multi-point
XSR(config-if<D1>)#encapsulation ppp
XSR(config-if<D1>)#dialer map ip 20.20.20.1 9051234567
```

ppp quality

This command sets the minimum Link Quality Monitoring (LQM) value on a serial interface before the link will go down.

Percentages are calculated for both incoming and outgoing directions. The outgoing quality is calculated by comparing the total number of packets and bytes sent to the total number of packets and bytes received by the destination node. The incoming quality is calculated by comparing the total number of packets and bytes received to the total number of packets and bytes sent by the destination node.

If the link quality percentage is not maintained, the link is considered of poor quality and taken down (by sending a DOWN event to all active NCPs). LQM forces a time lag so the link does not bounce up and down.

Syntax

ppp quality [*percentage*]

percentage	Sets the link quality threshold, ranging from 1 to 100.

Syntax of the "no" Form

Use the *no* form of this command to disable LQM:

```
no ppp quality
```

Default

Disabled

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example enables LQM on Serial interface *2/0*:

```
XSR(config)#interface serial 2/0
XSR(config-if<S2/0>)#encapsulation ppp
XSR(config-if<S2/0>)#no shutdown
XSR(config-if<S2/0>)#ppp quality 75
```

ppp timeout retry

This command sets the restart timer for Configure-Requests and Terminate-Requests on a Point-to-Point interface. The timer is the peak interval to wait for a response during PPP negotiation. This command applies to any serial port on which PPP encapsulation is enabled.

Syntax

ppp timeout retry *seconds*

seconds	Restart timer interval, ranging from 1 to 255 seconds.

Syntax of the "no" Form

The *no* command resets the timer to the default value:

```
no ppp timeout retry
```

Default

3

Mode

Serial, Dialer, and Fast/GigabitEthernet Sub-interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example resets the restart timer of Serial interface *1*:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1>)#encapsulation ppp
```

```
XSR(config-if<S1>)#ppp timeout retry 20
```

username

This command adds or modifies a user who can manage the XSR.

Note: Refer to "Network Management" on page 1 for more details.

This command specifies the password to be used in the PPP Challenge Handshake Authentication Protocol (CHAP) caller identification and by the Password Authentication Protocol (PAP).

A username entry is **required** for each remote system that the XSR communicates with and from which it seeks authentication for protocols such as CHAP and PAP or MSCHAP. When the XSR receives CHAP and MSCHAP challenges, the received username is searched through the list of usernames to find a password so it can send a response.

When the XSR receives responses to its challenges, the response name is searched through the list of usernames and passwords and compared. When the XSR receives PAP responses it also searches through its list of usernames to match passwords.

Syntax

username *name* **password** {**cleartext** | **secret** *type*} *password*

name	User ID.
cleartext	The password will not be encrypted.
secret	The password will be encrypted.
type	**0** or **5**. 0 means the input password is expected to be unencrypted; 5 means the input password is already encrypted so it will not be encrypted again.
password	For CHAP authentication: specifies the secret password for the local router or the remote system. The secret is encrypted when stored on the local router. The password can be up to 255 ASCII characters. Enclose the password in double quotes if entering a string with spaces. There is no limit to the number of username-password combinations that can be specified, allowing any number of remote systems to be authenticated.

Syntax of the "no" Form

The *no* form of this command deletes the user:

no username *name*

Default

No password is predefined

Mode

Global configuration: **XSR(config}#**

Example

The following example enables CHAP on serial interface *1/0* and defines a password for local server *Bob* and remote server *John*:

```
XSR(config)#hostname Bob
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation ppp
XSR(config-if<S1/0>)#ppp authentication chap
XSR(config)#username John password remote_dev
```

PPP Debug, Clear and Show Commands

debug ppp packet

This command enables PPP debugging for an interface from *outside* the actual interface. It performs the same PPP debugging as the **`ppp debug packet`** command but is issued from EXEC mode.

Note: All XSR debug commands are set to privilege level 15 by default.

Syntax

`debug ppp packet` [*interface type/number*] **`limit`** [*x*][*type1*][*type2*]…

interface type	Dialer, ATM, Serial, BRI, Multilink, or Fast/GigabitEthernet interfaces.
number	Interface number.
x	Total number of packets to debug, ranging from 1 to 1,000,000.
type1	Packet types to debug including: *PAP, CHAP, AUTH, BACP,*
type2	*BAP, BCP, CCP, ECP, IPCP, IPXCP, LCP* and *LQM*.

Syntax of the “no” Form

The following *no* form of the command returns the default value:

`no debug ppp packet` [*interface type/number*]

Mode

EXEC configuration: **`XSR>`**

Note: This command does not display in the *running config* file since it is strictly a debug function. It must be set manually every time you reboot the XSR.

Example

The following example debugs sets PPP debugging on Serial interface *2/0:0* with a limit of *10* packets for *LCP, BACP* and *BAP* protocols:

```
XSR#debug ppp packet serial 2/0:0 limit 10 lcp bacp bap
```

Sample Output

The following debugging output displays *all* PPP control packets:

```
May 21, 2003: 13:00:00 Rx 20 bytes LCP CONFIG_REQ:
MRU: 1500
Magic Number: 12345678 (0xBC614E)

May 21, 2003: 13:00:00 Tx 12 bytes IPCP CONFIG_ACK:
IP Address: 10.10.10.10

If the length field in the packet in the content does not match the total packet
length, it will be displayed as a warning:
May 21, 2003: 13:00:00 Rx 20 bytes LCP CONFIG_REQ:
MRU: 1500
Magic Number: 12345678 (0xBC614E)
(WARNING!!! NOT MATCHING PCK LENGTH 60bytes)
```

ppp debug packet

This command invokes debugging of Type 1 and 2 PPP control packets (transmit and receive) on Serial, Multilink, or Dialer interfaces. For Multilink, debugging is applied only to the bundle which handles IPCP and BAP/BACP negotiations. For Dialer interfaces, it is applied to the Serial interface that the dialer allocates to dial out. Within the control packet, the following fields are decoded and displayed: *protocol* (see list below), *code* (type of packet), *packet identifier*, *packet length*, and the *type*, *length* and *content* of the *option*.

You can select these packet types to be debugged: *PAP, CHAP, MS-CHAP, AUTH, BACP, BAP, BCP, CCP, ECP, IPCP, IPXCP, LCP, MLPPP,* and *LQM*. You can specify up to nine packets types to be debugged, and if you choose all packet types, entering **`ppp debug packet`** is sufficient. You can also choose to specify the same packet type repeatedly that is, **`ppp debug packet auth auth auth auth`**) which will have the same effect as issuing the packet type once.

Notes: You do not necessarily need to set a limit to be able to specify the types of packets. But, you cannot specify packet type *first* and then request a limit.

All XSR debug commands are set to privilege level 15 by default.

This command does not display in the running config file since it is strictly a debug function. It must be set manually every time you reboot the XSR.

You must issue this command after you enter encapsulation ppp.

Syntax

`ppp debug packet limit` *[x][type1][type2]...*

x	Total number of packets to debug, ranging from 1 to 1,000,000.
type1	Packet types to debug including: *PAP, CHAP, AUTH, BACP,*
type2	*BAP, BCP, CCP, ECP, IPCP, IPXCP, LCP* and *LQM*.

Syntax of the "no" Form

The *no* form of this command removes PPP debugging on the interface:

```
no ppp debug packet
```

Default

Limit: 100 packets

Mode

Interface configuration: **XSR(config-if<xx>)#**

Example

This example sets PPP debugging of *IPCP* and *LQM* packets with a *50*-packet limit on Serial *1/0*:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation ppp
XSR(config-if<S1/0>)#ppp debug packet limit 50 ipcp lqm
```

Sample Output

The following debugging output is displayed on Multilink interface 57:

```
XSR#show interface multilink 57
********** Multilink Interface Stats **********
Multilink 57 is Admin Up
Internet address is 192.168.34.1, subnet mask is 255.255.255.0
LCP          State: OPENED
IPCP         State: OPENED
Multilink    State: OPENED

Detailed Debug PPP Control Packet is ON for [type1] [type2] [type3], limit is [x],
number decoded is [y]
```

clear ppp

This command clears PPP counters for interfaces running PPP.

Syntax

```
clear ppp
```

Mode

Privileged EXEC: **XSR#**

Sample Output

The following output displays when you enter the **show ppp interface** command after clearing the serial *1/0:0* port:

```
XSR#show ppp interface

********** PPP Stats **********
Serial 1/0:0: PPP is Admin Up / Oper Up / Link Speed: 64000
LCP  Current State:                  OPENED
IPCP  Current State:                 OPENED
Multilink Current State:             OPENED

LCP STATS
Total Rcv Pck:                       0
Total Rcv Control Pck:               0
Total Rcv Data Pck:                  0
Total Rcv Pck Discarded:             0

Total Tx Pck:                        0
Total Tx Control Pck:                0
Total Tx Data Pck:                   0
Total Tx Pck Discarded:              0

Rx Control Pck Discarded:            0
Rx Control Pck Error:                0
Rx Control Pck Unknown protocol:     0
Rx Control Pck Too Long:             0

LocalToRemoteProtocolCompression:    Disabled
RemoteToLocalProtocolCompression:    Disabled
LocalMRU:                            1500
RemoteMRU:                           1500
ReceiveFcsSize:                      16
TransmitFcsSize:                     16

LQR STATS
No LQM Monitoring

LCP CONFIGURATION
InitialMRU:                          1500
MagicNumber:                         true
FcsSize:                             16
LQR CONFIGURATION
Period:                              10 sec
Status:                              Disabled
```

show ppp

This command displays all configured PPP ports and status including Link Control Protocol (LCP) and Link Quality Monitoring (LQM) states.

Syntax

```
show ppp
```

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following output is displayed for *Serial* and *Multilink* interfaces:

```
XSR#show ppp
Serial 1/0 PPP State:
   LCP State: OPENED   IPCP State: OPENED   Multilink State: OPENED
Multilink 8 MLPPP State:
   LCP State: OPENED   IPCP State: OPENED   Multilink State: OPENED
```

The following output is displayed for configured *Dialer* interfaces:

```
XSR#show ppp
Dialer0   LCP Current State:
   INITIAL IPCP Current State: INITIAL
Dialer1 MLPPP State:
      LCP State: opened Multilink State: opened
Dialer2 MLPPP State:
      LCP State: opened Multilink State: opened
Dialer3 MLPPP State:
      LCP State: opened Multilink State: opened
Dialer4 MLPPP State:
      LCP State: opened Multilink State: opened
Dialer5 MLPPP State:
      LCP State: opened Multilink State: opened
Dialer33 MLPPP State:
      LCP State: opened Multilink State: opened
Dialer44 MLPPP State:
      LCP State: opened Multilink State: opened
Dialer1 MLPPP State:
      LCP State: opened Multilink State: opened
Multilink 4 MLPPP State:
      LCP State: opened Multilink State: opened
```

show interface serial

This command displays interface statistics and PPP status if the interface is encapsulated with PPP.

Syntax

show interface [*card/port:channel number*] [*type | type number*]

card/port	The PPP WAN port for which to view link status, stats and configuration data.
type	Serial or Dialer - Interface types which PPP can run on.
number	Card/port - for serial interface. Card/port:channel number - for serial channel groups. Number - for other logical interfaces such as Dialer.

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following output is produced by this command:

```
Serial 1/0 is Admin Up / Oper Up
Internet address is 25.25.25.3, subnet mask is 255.255.255.0
LCP       State: OPENED
IPCP      State: OPENED
Multilink State: OPENED
```

show ppp interface

This command displays all configured PPP instances, the interface they belong to and their status.

To issue this command correctly, follow the guidelines below:

- Issuing the **show ppp interface** command without any other parameter displays link status, statistics and configuration for all interfaces running PPP.
- The **show ppp interface** *type* command displays link status, statistics and settings for any interface *type* running PPP.
- The **show ppp interface** *type number* command displays link status, statistics and configuration for the interface type *number*.
- The **show ppp interface** *dialer number* [*multi-class serial*] command displays Dialer statistics with Serial and Multiclass options.
- The **show ppp interface** *multilink number* [*bap | memberlink | multi-class*] command displays multilink statistics with various options.

Syntax

show ppp interface *card/port* [*type number options*]

card/port	The NIM number and PPP WAN port:channel number to view associated link status, statistics and settings.
type	The interface type PPP is running on including: *Dialer (0* to *255), Multilink (1* to *32767),* or *Serial* (see below).
number	Card/port numbers or Card/port:channel number.
option	memberlink, mlpppgroup (MLPPP only), multi-class, or bap (MLPPP only) statistics.

The Serial port card, port, sub-interface, and channel numbers are expressed as follows:

0	Console port.
<1-2>/<0-3>	Card and port number.
<1-2>/<0-3>.<1-30>	Card, port, sub-interface number.
<1-2>/<0-3>:<0-31>	Card, port and channel number.
<1-2>/<0-3>:<0-31>.<1-30>	Card, port, channel and sub-interface number

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following output displays with a PPP connection established (PPP quality has not been enabled on the interface so the LINK QUALITY statistic is not monitoring):

```
XSR>show ppp interface serial 1/0

********** MLPPP Stats **********
Multilink 8: MLPPP is Admin Up / Oper Up
Group Num: 8
LCP       State: OPENED
IPCP      State: OPENED
Multilink State: OPENED

Bundle Size:            31
Max Load Threshold:     120
Bundle Tx Load Avg:     240
Bundle Rx Load Avg:     240
Last Tx  Seq Num:       14787652
Last Fwd Seq Num:       12933548
Last Rcv M:             12933518
No Of Frag Rcvd:        12920875
No Of Frag Discard:     0
No Of Frag in Rcv List: 11
No Of Pck  in Tx Buf Q: 0
Reassem Start Tick:     3882798
Last M Change Tick:     3882815
High Pri Member link is Serial 1/0:29

Multilink PPP includes following memberlink interface:
Serial 1/0:2
Serial 1/0:6
Serial 1/0:9
Serial 1/0:15
Serial 1/0:17
Serial 1/0:18
Serial 1/0:19
Serial 1/0:23
Serial 1/0:26
Serial 1/0:28
Serial 1/0:30
Serial 1/0:20
Serial 1/0:27
Serial 1/0:22
Serial 1/0:21
Serial 1/0:8
Serial 1/0:4
```

```
Serial 1/0:0
Serial 1/0:3
Serial 1/0:7
Serial 1/0:13
Serial 1/0:10
Serial 1/0:1
Serial 1/0:25
Serial 1/0:11
Serial 1/0:24
Serial 1/0:12
Serial 1/0:5
Serial 1/0:16
Serial 1/0:14
Serial 1/0:29
```

The following displays output with PPP quality enabled and a PPP connection:

```
XSR>show ppp serial 0/4/1

********** PPP Stats **********
Interface Serial 0/4/1

LCP  Current State:                 OPENED
IPCP  Current State:                OPENED
Multilink Current State:            OPENED

LCP STATS
Total Rcv Pck:                      1618575
Total Rcv Control Pck:              420
Total Rcv Data Pck:                 1618155
Total Rcv Pck Discarded:            1

Total Tx Pck:                       1618653
Total Tx Control Pck:               420
Total Tx Data Pck:                  1618233
Total Tx Pck Discarded:             2

Rx Control Pck Discarded:           0
Rx Control Pck Error:               0
Rx Control Pck Unknown protocol:    0
Rx Control Pck Too Long:            0

LocalToRemoteProtocolCompression:   Disabled
RemoteToLocalProtocolCompression:   Disabled
LocalMRU:                           1500
RemoteMRU:                          1500
ReceiveFcsSize:                     16
TransmitFcsSize:                    16

LQR STATS
```

```
Quality:         good   InGoodOctets:  26600
LocalPeriod:     100000    RemotePeriod:   100000
OutLQRs:1000InLQRs: 1000

LCP Configuration:

LCP CONFIGURATION
InitialMRU:                          1500
MagicNumber:                         true
FcsSize:                             16
LQR CONFIGURATION
Period:                              10 sec
Status:                              Disabled
```

Output Parameters Summary

For PPP link status and statistics, refer to the following section. For *LQR status* and *statistics*, go to page 106. For *LQR parameters*, go to page 107.

LCP Statistics

This section displays PPP-link specific management information.

Rx Control Pck Discarded

Range	32-bit counter
Description	Sum of received packets discarded because length is too short (less than 4).

Rx Control Pck Error

Range	32-bit counter
Description	Sum of received packets n detected with an error in the control field.

Rx Control Pck Unknown protocol

Range	32-bit counter
Description	Sum of received packets detected with an unknown protocol field.

Rx Control Pck Too Long

Range	32-bit counter
Description	Sum of received packets discarded because their length exceeded the MRU. Packets that are longer than the MRU but which are successfully received and processed are NOT included in this count.

LocalToRemoteProtocolCompression

Range	INTEGER {enabled (1), disabled (2)}
Description	Indicates whether the local PPP entity will use Protocol Compression when sending packets to the remote PPP entity. The value is meaningful only when the link has reached the open state.

RemoteToLocalProtocolCompression

Range	INTEGER {enabled (1), disabled (2)}
Description	Indicates whether the remote PPP entity will use Protocol Compression when sending packets to the local PPP entity. The value is meaningful only when the link has reached the open state.

LocalMRU

Range	INTEGER (1…2147483648)
Description	Current value of the MRU for the local PPP Entity. This value is the MRU that the remote entity uses when sending packets to the local PPP entity. The value is meaningful only when the link has reached the open state.

RemoteMRU

Range	INTEGER (1...2147483648)
Description	Current value of the MRU for the remote PPP Entity. This value is the MRU that the local entity uses when sending packets to the remote PPP entity. The value is meaningful only when the link has reached the open state.

ReceiveFcsSize

Range	INTEGER (0...128)
Description	Size of the Frame Check Sequence (FCS) in bits that the remote node will generate when is sending packets to the local node. The value is meaningful only when the link has reached the open state.

TransmitFcsSize

Range	INTEGER (0...128)
Description	Size of the Frame Check Sequence (FCS) in bits that the local node will generate when is sending packets to the remote node. The value is meaningful only when the link has reached the open state.

LQR Status and Statistics

This section displays LQR parameters displayed for the local PPP entity. Values are displayed only if LQR Quality Monitoring has been successfully negotiated on the link.

Quality

Range	Integer - Good, Bad, or Not-determined
Description	Current quality of the link as declared by the local PPP entity's Link Quality Management modules. No effort is made to define good or bad, nor is the policy used to learn it. The not-determined value indicates that the entity does not actually evaluate the link's quality. This value clarifies the *determined to be good case* from the *node termination made and presumed to be good* case.

LocalPeriod

Range	Integer - 1 to 2147483648

Description	The LQR reporting period, in hundredths of a second, that is in effect for the local PPP entity.

OutLQRs

Range	32-bit counter
Description	Value of the OutLQRs counter on the local node for the link. OutLQRs increases by one for each transmitted Link -Quality -Report packet.

LCP Configuration

This section describes LCP configuration data displayed for a PPP Link.

InitialMRU

Range	Integer - 0 to 2147483647
Description	Initial Maximum Receive Unit (MRU) that the local PPP entity will advertise to the remote entity. If the value of this variable is 0 then the local PPP entity will not advertise any MRU to the remote entity and the default MRU will be assumed. Changing this object will take effect when the link is next restarted.
Default	1500

MagicNumber

Range	Integer - False or True
Description	If true (2), the local node will try to perform Magic Number negotiation with the remote node. If false (1), negotiation is not tried. The local node will comply with any magic number negotiations tried by the remote node, per the PPP RFC. Changing this object will take effect when the link is next restarted.
Defaults	False

FcsSize

Range	Integer - 0 to 128
Description	Size of the FCS, in bits, the local node will try to negotiate for use with the remote node. Regardless of this value's object, the local node will comply with any FCS size negotiations started by the remote node, according to the PPP RFC. Changing this object will take effect when the link is next restarted.
Default	16

LQR Configuration

This section describes LQR configuration data displayed for a PPP link.

Period

Range	Integer - 0 to 2147483647
Description	The LQR Reporting Period that the local PPP entity will attempt to negotiate with the remote entity, in hundredths of a second. Changing this object will take effect when the link is next restarted.
Default	0

Status

Range	Integer - Disabled or Enabled
Description	If enabled(2), the local node will try to perform LQR negotiation with the remote node. If disabled(1), negotiation is not tried. The local node will comply with any magic number negotiations tried by the remote node, according to the PPP RFC. Changing this object takes effect when the link is next restarted.
Default	Enabled

Multilink PPP Commands

interface multilink

This command names the multilink group and creates a logic interface for this multilink group. Only the PPP multilink group is supported currently.

Syntax

interface multilink *number [1-32767]*

1-32767	Designation of the virtual multilink group.

Syntax of the "no" Form

The *no* form of this command deletes the multilink group:

no interface multilink *number [1-32767]*

Default

No multilink group

Mode

Global configuration: **XSR(config)#**

Next Mode

Multilink Interface configuration: **XSR(config-if<Mxx>)#**

Example

The following example enables multilink on group 2 with serial interface *1/1* configured as the physical interface:

```
XSR(config)#interface multilink 2
XSR(config-if<M2>)ppp multilink endpoint ip 192.168.10.214
XSR(config-if<M2>)ip address 192.168.10.213 255.255.255.252
XSR(config-if<M2>)no shutdown

XSR(config)#interface serial 1/1
XSR(config-if<S1/1>)#media-type X21
```

```
XSR(config-if<S1/1>)#multilink-group 2
XSR(config-if<S1/1>)#encapsulation ppp
XSR(config-if<S1/1>)#ppp multilink
XSR(config-if<S1/1>)#no shutdown
```

multilink max-links

This command sets the maximum number of links allowed in this bundle. If multilink BAP is configured and the number of active links exceed the maximum number of links, BAP will try to negotiate the links down.

Syntax

multilink max-links *number (1-255)*

1-255	Maximum number of links allowed in this bundle.

Default

16

Mode

Dialer Interface configuration: **XSR(config-if<xx>)#**

Example

This example sets the minimum multilink limit to *6* on Dialer port *4*:

```
XSR(config)#interface dialer 4
XSR(config-<D4>)#multilink min-links 6
```

multilink min-links

This command triggers the dialer to maintain the minimum number of links in a bundled multilink over a switched line and should be configured on the *called* side of a connection. It is the first means by which the XSR effects Bandwidth-on-Demand (BoD).

The **multilink load-threshold** command is the second means by which the XSR controls traffic via BoD. A third means to effect BoD is by use of the Bandwidth Allocation Protocol (BAP) which is activated by several **ppp bap** commands. BAP negotiates with the peer to add or drop a link, and can request a phone number from a central repository with the **ppp bap number** command. If multilink BAP is configured and the number of active links is less than the minimum number of links, BAP will try to negotiate the links up.

Syntax

multilink min-links *number (1-255)*

1-255	Minimum number of links allowed in this bundle.

Default

1

Mode

Dialer Interface configuration: **XSR(config-if<*xx*>)#**

Examples

The following example sets the minimum multilink limit to *6* on the terminating dialer interface:

```
XSR(config)#interface dialer 4
XSR(config-if<D4>)#multilink min-links 6
```

ppp bap call

This command sets Bandwidth Allocation Protocol (BAP) call values on a dialer interface to set up Bandwidth-on-Demand (BoD). It permits the port to accept links *from* and initiate links *to* a peer.

The **multilink load-threshold** command is a second means by which the XSR controls traffic via BoD. It is also provided by setting the **multilink min-links** command.

Note: The **multilink load-threshold** command must be set to operate BAP.

Syntax

ppp bap call {*accept* | *request*}

accept	Accepts links from a peer. This default lets peers can add links to the ML bundle.
request	Lets the local side of the connection start links. Set up on the *called* side of a link only.

Syntax of the “no” Form

The *no* form of this command disables previously set BAP values:

no ppp bap call {*accept* | *request*}

Default

Accept

Mode

Dialer Interface configuration: **XSR(config-if<D*x*>)#**

Example

The following example sets BAP call values on Dialer interface *57*:

```
XSR(config)#interface dialer 57
XSR(config-if<D57>)#encapsulation ppp
XSR(config-if<D57>)#no shutdown
```

```
XSR(config-if<D57>)#ppp bap call accept
```

ppp bap callback

This command sets enables Bandwidth Allocation Protocol (BAP) callback parameters on a dialer interface to set up Bandwidth-on-Demand (BoD). It permits the port to initiate adding a link *to* or requesting a link *from* a peer. It applies to Dialer interfaces only.

The **multilink load-threshold** command is a second means by which the XSR controls traffic via BoD. It is also provided by setting the **multilink min-links** command.

Note: You must configure **multilink load-threshold** to run BAP.

Syntax

ppp bap callback {*accept* | *request*}

accept	Local router initiates a link addition upon peer notification.
request	Local router requests a peer to initiate a link.

Mode

Dialer Interface configuration: **XSR(config-if<D*x*>)#**

Mode of the “no” Form

The *no* form of this command removes callback configuration:

no ppp bap callback {*accept* | *request*}

Example

The following example configures BAP to accept and request callbacks:

```
XSR(config)#interface dialer 1
XSR(config-if<D1>)#encapsulation ppp
XSR(config-if<D1>)#no shutdown
XSR(config-if<D1>)#ppp bap callback accept
XSR(config-if<D1>)#ppp bap callback request
```

ppp bap number

This command specifies the Bandwidth Allocation Protocol (BAP) phone number which a peer can dial to connect and set up Bandwidth-on-Demand (BoD). It applies to dialer interfaces only.

The **multilink load-threshold** command is a second means by which the XSR controls traffic via BoD. It is also provided by setting the **multilink min-links** command.

Note: The **multilink load-threshold** command must be set to operate BAP.

Syntax

ppp bap number {**default** *phone-number*}

default *phone-number*	Primary number for incoming calls. Up to 5 numbers can be entered.

Syntax of the "no" Form

The *no* command removes a BAP phone number:

no ppp bap number {**default** *phone-number*}

Mode

Dialer Interface configuration: **XSR(config-if<D*x*>)#**

Example

The following example specifies the BAP *default* phone number:

```
XSR(config)#interface dialer 1
XSR(config-if<D1>)#ppp bap number
```

ppp bap timeout

This command configures Bandwidth Allocation Protocol (BAP) action timeouts to set up Bandwidth-on-Demand (BoD).

The **multilink load-threshold** command is a second means by which the XSR controls traffic via BoD. It is also provided by setting the **multilink min-links** command.

Syntax

ppp bap timeout {**pending** *seconds* | **response** seconds}

pending *seconds*	Wait interval for pending actions. Range: 2 to 180 seconds.
response *seconds*	Wait interval for response packets. Range: 2 to 180 seconds.

Syntax of the "no" Form

The *no* command deletes BAP action timeouts:

no ppp bap timeout {**pending** | **response**}

Defaults

- Pending seconds: 20
- Response seconds: 20

Mode

Dialer Interface configuration: **XSR(config-if<Dx>)#**

Example

The following example resets the BAP pending timeout on Dialer port 1:

```
XSR(config)#interface dialer 1
XSR(config-if<D1>)#ppp bap timeout pending 60
```

ppp multilink

This command enables Multilink PPP on an XSR interface. Multilink PPP operates over single or multiple interfaces that are configured to support both Dial-on-Demand rotary groups and PPP encapsulation. It applies to asynchronous serial interfaces, and ISDN leased-line Basic Rate Interfaces (BRIs), and ISDN Primary Rate Interfaces (PRIs).

This command is associated with the following multilink sub-commands:

- **`endpoint`** sets the multilink group Endpoint Descriptor over the multilink bundle. Refer to page 8-114 for command details.
- **`fragment-delay`** sets the maximum fragment delay interval. Refer to page 8-115 for command details.
- **`fragment disable`** disables fragmentation over a multilink PPP connection. Refer to page page 8-117 for command details.
- **`group`** - configures a PPP link and assigns it to a specified PPP multilink group. Refer to page 8-118 for command details.
- **`load-threshold`** set the value which triggers the dialer to add or delete a link from the multilink bundle. See page 8-119 for details.
- **`multi-class`** sets the Multi-Class MLPPP option for the MLPPP header format. Refer to page 8-120 for command details.

Multilink PPP BAP is designed to manage bandwidth of a multilink bundle. BAP works in conjunction with the **`multilink load-threshold`** command to enable Bandwidth-on-Demand (BoD) when bandwidth must be added or removed on the XSR.

BAP negotiates with the peer to add or drop a link, and can request a phone number from a central site repository using the **`bap number default`** command.

Note: BAP is employed on Dialer and ISDN lines only.

Use the **`multilink load-threshold`** command to enable a dialer interface (dialer profile) to bring up additional links and add them to a multilink bundle. If you want a multilink bundle to be connected *indefinitely*, you must set a very high idle timer.

Syntax

`ppp multilink` {*bap*}

bap	Enables BAP/BACP to be negotiated over the multilink bundle.

Syntax of the “no” Form

The *no* form of this command not only removes multilink on the interface but also multilink BAP if it also was configured:

`no ppp multilink` {*bap*}

Default

Disabled

Mode

Dialer or Serial Interface configuration: **XSR(config-if<D/S*xx*>)#**

Examples

The following example configures a dialer for Multilink PPP. It does not show the configuration of the physical interfaces.

```
XSR(config)#interface dialer 0
XSR(config-if<D0>)#ip address 101.0.0.2 255.0.0.0
XSR(config-if<D0>)#encapsulation ppp
XSR(config-if<D0>)#dialer idle-timeout 500
XSR(config-if<D0>)#dialer map ip 101.0.0.1 name ny broadcast 41612345678922
XSR(config-if<D0>)#dialer load-threshold 30 either
XSR(config-if<D0>)#ppp authentication chap
XSR(config-if<D0>)#ppp multilink
```

The following example configures Multilink PPP leased-line service on BRI interface *2/1*. Specifying the leased-line speed of *56* kbps adds two B-channels to the BRI port, one of which is enabled for Frame Relay service.

```
XSR(config)#interface bri 2/1
XSR(config-if<BRI-2/1>)#leased-line 56
XSR(config)#interface bri 2/1:1
XSR(config-if<BRI-2/1:1>)#encapsulation ppp
XSR(config-if<BRI-2/1:1>)#ppp multilink
XSR(config-if<BRI-2/1:1>)#ppp multilink group 1
XSR(config)#interface bri 2/1:2
XSR(config-if<BRI-2/1:2>)#ip address 3.3.3.4 255.255.255.0
XSR(config-if<BRI-2/1:2>)#encapsulation frame-relay
XSR(config)#interface multilink 1
XSR(config-if<M1>)#ip address 3.3.3.3 255.255.255.0
```

ppp multilink endpoint

This command sets the multilink group Endpoint Descriptor (EPD) value (class) over the multilink bundle. It applies only to interfaces that can configure a bundle interface including Multilink, Dialer, and ISDN BRI or PRI interfaces.

Syntax

ppp multilink endpoint [*null* | *hostname* | *ip_address* | *mac interface* | *fastethernet (1-2) string* | *phone*]

null	NULL class is specified with a value of 0.
hostname	Local Assigned address class is set with a local host name entered using the **hostname** command.
ip_address	IP address class is set with a specified IP address value.

mac interface	IEEE 802.1 Global MAC address class is set with a MAC address of either Fastethernet 1 or 2. *fastethernet*
string	PPP Magic Number class is specified. Instead of using the negotiated PPP magic number, you can specify any string less than 20 characters.
phone	PSTN Directory Number class set with a phone number of no more than 15 digits.

Mode

Dialer, Multilink, BRI Interface, and Controller configuration: **XSR(config-if<*xx*>)** and **XSR(config-controller<T/E*xx*>)**

Default

Hostname

Example

The following example sets the PPP multilink endpoint value over virtual multilink interface *57*:

```
XSR(config)#interface multilink 57
XSR(config-if<M57>)#ppp multilink endpoint null
XSR(config-if<M57>)#ppp multilink endpoint hostname
XSR(config-if<M57>)#ppp multilink endpoint ip address 1.1.1.1
XSR(config-if<M57>)#ppp multilink endpoint string aaaaaaa
XSR(config-if<M57>)#ppp multilink endpoint phone 1234567890
```

ppp multilink fragment-delay

This command sets the maximum fragment delay interval in milliseconds. The value is used to compute the maximum fragment size that can be sent over each member link in the bundle. The maximum fragment size is calculated as:

Fragment size (in bytes) = fragment-delay (ms) x link speed (kbps) / 8

Note: The maximum fragment size is limited to 1500 bytes.

Table 8-1 below shows the relationship between maximum fragment delay and maximum fragment size. Italicized figures indicate *bytes*.

Each MLPPP packet includes one fragment with an additional HDLC header (2 bytes), PID (2 bytes), MLPPP header (2/4 bytes for short/long sequence number format) and FCS (2 bytes).

The actual fragment size will be decided after the load balance over member link is taken into account and should not exceed the maximum fragment size allowed. When the command is

entered, no maximum fragment size will be set and the fragment size will only be decided with the load balance.

Table 8-1 Maximum Fragment Size (bytes)/Fragment Delay (ms)

Link Speed	Fragment Delay (ms)						
	5 ms	10 ms	20 ms	50 ms	100 ms	500 ms	1000 ms
56 kbps	*35*	*70*	*140*	*280*	*560*	*1120*	*1500*
64 kbps	*40*	*80*	*160*	*320*	*640*	*1280*	*1500*
128 kbps	*80*	*160*	*320*	*640*	*1280*	*1500*	*1500*
256 kbps	*160*	*320*	*640*	*1280*	*1500*	*1500*	*1500*
512 kbps	*320*	*640*	*1280*	*1500*	*1500*	*1500*	*1500*
768 kbps	*640*	*1280*	*1500*	*1500*	*1500*	*1500*	*1500*
1536 kbps	*1280*	*1500*	*1500*	*1500*	*1500*	*1500*	*1500*
2024 kbps	*1500*	*1500*	*1500*	*1500*	*1500*	*1500*	*1500*

Syntax

ppp multilink fragment-delay *value*

value	Delay interval ranging from 10 to 1000 in milliseconds.

Syntax of the “no” Form

The *no* form of this command deletes the fragment-delay setting:

no ppp multilink fragment-delay

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

10 milliseconds

Example

The following example sets the fragment-delay to *30* milliseconds on the Dialer 2 interface:

```
XSR(config-if<D2>)#ppp multilink fragment-delay 30
```

ppp multilink fragment disable

This command disables fragmentation over a bundle PPP connection, supporting Multilink and Dialer interfaces.

Syntax

```
ppp multilink fragment disable
```

Syntax of the “no” Form

The no form of this command enables fragmentation (default mode):

```
no ppp multilink fragment disable
```

Mode

Interface configuration: **XSR(config-if<xx>)#**

Default

Enabled

Examples

The following example disables fragmentation over Multilink interface *1*:

```
XSR(config-if<M1>)#ppp multilink fragment disable
```

Display Examples

The following examples display fragmentation settings by the **show interface multilink** command:

```
XSR#show interface multilink 1
********** Multilink Interface Stats **********
Multilink 1 is Admin Up
Internet address is 30.30.30.2, subnet mask is 255.255.255.0
LCP         State: OPENED
IPCP        State: OPENED
Multilink   State: OPENED
Multi-Class State: OPENED

Multilink header format is LONG SEQ NUM
Class suspendable level is 5 tx classes and 5 rcv classes

Fragmentation is disabled
```

.....

The following example displays fragmentation settings by the **show ppp interface multilink** command:

```
XSR#show ppp interface multilink 1
********** MLPPP Bundle Stats **********
Multilink 1: MLPPP is Admin Up / Oper Up
```

```
Group Num: 1
LCP          State: OPENED
IPCP         State: OPENED
Multilink    State: OPENED
Multi-Class State: OPENED

Multilink header format is LONG SEQ NUM
Class suspendable level is 5 tx classes and 5 rcv classes

Fragmentation is disabled
Bundle Size:             2
Class Level Tx:          5
            Rx:          5
Max Load Threshold:      0
Bundle Tx Load Avg:      0
Bundle Rx Load Avg:      0
No Of Pck  in Rx Buf Q: 0
Lowest link Speed:       1536000
Max Fragment Size:       Not Set
High Pri Member link is Serial 2/0:0
.....
```

The following example displays fragmentation settings:

```
XSR# show ppp interface multilink 1 multiclass
********** MLPPP Bundle MultiClass Stats **********
Multilink 1: MLPPP is Admin Up / Oper Up
Group Num: 1
LCP          State: OPENED
IPCP         State: OPENED
Multilink    State: OPENED
Multi-Class State: OPENED

Multilink header format is LONG SEQ NUM
Class suspendable level is 5 tx classes and 5 rcv classes

Fragmentation is disabled
Max Fragment Size is not set
```

ppp multilink group

This command configures a PPP link and assigns it to or removes it from a specified PPP Multilink bundle. It applies only to interfaces that can configure a bundle interface including multilink, dialer, and ISDN BRI or PRI interfaces.

Syntax

multilink group *1 - 32767*

1 - 32767	Designation of the PPP multilink group.

Syntax of the "no" Form

The *no* form of this command removes the PPP multilink group:

```
no multilink-group
```

Default

Disabled with no specific multilink group assigned

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Examples

The following example assigns PPP link Serial interface *1/1* to the PPP multilink group *20*:

```
XSR(config-if<S1/1>)#multilink group 20
```

The next example *also* assigns PPP link Serial interface *1/1* to the PPP multilink group *20*:

```
XSR(config-if<S1/1>)#ppp multilink group 20
```

multilink load-threshold

This command sets the multilink load threshold which triggers the dialer to add or delete a link from the multilink bundle. It should be configured on the *called* side of a connection only. This command effects Bandwidth-on-Demand (BoD) on the XSR.

In determining whether to trigger the dialer, the XSR monitors only the bundle load. The load threshold provides the dialer with a trigger to add or delete the multilink member link from the member link bundle. The load is sampled every second and averaged over an 8-second period. Triggering is delayed for 10 seconds when the load surpasses or falls below the threshold.

Triggering is generated when:

- Either the inbound or outbound traffic surpasses the threshold; or
- Both inbound and outbound traffic fall below the threshold.

No triggering is generated when:

- The number of member links is already equal to the *max-links* value set on the bundle when the load surpasses the threshold; and
- The number of the links is already equal to the *min-links* value set on the bundle when the load falls below the threshold.

The **multilink load-threshold** command is the second means by which the XSR controls traffic via BoD. It is also provided by setting the **multilink min-links** command, which is the first means by which the XSR controls traffic. A third means used to effect BoD is by use of the Bandwidth Allocation Protocol (BAP) which is activated by several **ppp bap** commands. BAP negotiates with the peer to add or drop a link, and can request a phone number from a central repository with the **ppp bap number** command.

Note: To avoid unexpected behavior, configure this command on one peer only. If it is set on both peers, their threshold values should match.

Syntax

multilink load-threshold *number (1-255)*

1-255	Load on the port: 255 indicates it has reached 100% of bandwidth.

Default

255

Mode

Dialer Interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example sets the multilink PPP load threshold to *250* on the terminating Dialer interface:

```
XSR(config)#interface dialer 4
XSR(config-<D4>)#multilink load-threshold 250
```

ppp multilink multi-class

This command enables Multi-Class MLPPP (Multilink PPP) for the Multilink PPP header format providing Quality of Service (QoS) for selected packets between peers. It supports five streams of sequence numbers, the long sequence format by default, and the short sequence number by negotiation. Any class lower than the default requested by the peer will be accepted, and higher than the default will eventually trigger a reject message if the value is accepted by the peer.

Syntax

ppp multilink multi-class

Syntax of the “no” Form

The *no* form of this command disables multi-class MLPPP:

no ppp multilink multi-class

Defaults

- Long sequence number
- Accept negotiation for short sequence number
- Accept any suspendable (class) level less than or equal to 5
- Disabled

Mode

Dialer or Multilink Interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example enables the multi-class MLPPP option:

```
XSR(config-if<D57>)#ppp multilink multi-class
```

Multilink Show Commands

show interface multilink

This command displays multilink interface statistics including MLPPP status for both the bundle and the member link.

Syntax

show interface multilink [*number*]

card/port	The ML interface port for viewing link status, statistics and configuration data.
number	Logical interfaces.

Mode

EXEC: **XSR>**

Sample Output

Thefollowing is sample output for Multilink interface *8*:

```
XSR>show interface multilink 8
********** Multilink Interface Stats **********
Multilink 8: MLPPP is Admin Up / Oper Up
Group Num: 8
LCP           State: OPENED
IPCP          State: OPENED
Multilink     State: OPENED
Multi-Class   State: OPENED

Multilink header format is LONG SEQ NUM
Class suspendable level is 5 tx classes and 5 rcv classes

Max Fragment delay is 10 ms
MLPPP Bundle Info:
Control Object state is Admin Down / Oper Down
Multilink PPP has no memberlinks

Data Object state is Admin Down
The adjacent is DOWN and data passing is FALSE
Bundle size is 0
Max Load Threshold: 0
Total Load Bandwidth is 64000 bits/sec
Bundle Stats
Rx: Total          0, TX: Total          0
    Data           0,     Data           0
    Ctrl           0,     Ctrl           0
    Null           0,     Null           0
    Drop           0,     Drop           0
Rx Load BW Avg   0, Max   0, Min   0
Tx Load BW Avg   0, Max   0, Min   0
```

PPP Multilink Status

LCP State

Range	INITIAL/ STARTING/ CLOSED/ STOPPED/ CLOSING/ STOPPING/ REQSENT/ ACKRCVD/ ACKSENT/ OPENED
Description	LCP state. Refer to RFC-1661 for details.

IPCP State

Range	INITIAL/ STARTING/ CLOSED/ STOPPED/ CLOSING/ STOPPING/ REQSENT/ ACKRCVD/ ACKSENT/ OPENED
Description	IPCP state. Refer to RFC-1332 for details.

Multilink State

Range	OPENED/CLOSED
Description	MLPPP state, OPENED if negotiation with peer successful; CLOSED otherwise.

Multi-Class State

Range	OPENED/CLOSED
Description	Multi-Class state, OPENED if negotiation is successful with the peer; CLOSED otherwise.

Bundle Size

Range	1-256
Description	Number of member links under the bundle.

Class Level Tx/Rx

Range	1-5
Description	Multi-Class level after negotiation. 1 for multi-class disabled.

Max Load Threshold

Range	0-255
Description	Zero (0) indicates load threshold monitoring is disabled.

Bundle Tx/Rx Load Avg

Range	0-255
Description	Average loading of Tx/Rx loading. 255 = 100% loading against the bandwith.

No Of Pck in Rx Buf Q

Range	Not defined.
Description	Number of packets in the rx forwarding buffer.

Lowest link Speed

Range	Not defined.
Description	Lowest speed link under the bundle for calculating the maximum fragment size.

Max Fragment Size

Range	Not defined.
Description	Maximum fragment size over the member links.

High Pri Member link is Serial 1/00

Range	Not defined.
Description	Highest speed link under the bundle. Used to transmit the control packet.

PPP Multilink Bundle Statistics

Rx Stats

Total	Sum of packets received under the bundle including data, control, Null content packet and the discarded packet.
Data	Sum of data packets received under the bundle.
Control	Sum of control packets received under the bundle.
Null	Sum of Null content packets received under the bundle, used for synchronizing tx/rx sequence number.
Discard Pck Too Long	Sum of packets discarded because size is too long, up to 1504 bytes.
Invalid Proto	Sum of packets discarded because protocol field is invalid for PPP.
Wrong Proto	Sum of packets discarded because protocol field is wrong for MLPPP.
Padding Error	Sum of packets discarded because padding size is wrong.
Invalid Cls#	Sum of packets discarded because class number greater than class level negotiated.
Error to CP	Sum of internal messages lost.
No Lower Lyr	Sum of packets discarded because lower layer is not ready.
No Upper Lyr	Sum of packets discarded because upper layer is not ready.
Others	Sum of packets discarded due to errors recorded in classes or member links.

Tx Stats

Total	Sum of packets transmitted under the bundle including data, control, Null content, and discarded packets.
Data	Sum of data packets transmitteded under the bundle.
Control	Sum of control packets transmitted under the bundle.
Null	Sum of Null content packets transmitted under the bundle. Used for synchronizing the tx/rx sequence number.
Discard Pck Too Long	Sum of packets discarded because the size is too long, up to 1504 bytes..
No Lower Lyr	Sum of packets discarded because the lower layer is not ready.
EnQueue Full	Sum of packets discarded because the transmission queue is full.
Others	Sum of packets discarded due to error recorded in classes or member links.

show ppp interface multilink/dialer

This command displays PPP status, statistics and configuration data for interfaces running PPP.

Syntax

show ppp interface [**interface** *type/number*][**option type**]

interface *type*	Dialer or multilink interface upon which MLPPP can be configured
number	Designation for multilink or dialer interface.
option type	Available options including the following:
none	Display general MLPPP status and statistics.
multi-class	Display Multi-Class related information.
bap	Display BAP-related information.
memberlink	Display multilink member link-related information

Mode

EXEC: **XSR>**

Sample Output

The following example displays output *without* Multi-Class configured:

```
********** MLPPP Bundle Stats **********
Multilink 8: MLPPP is Admin Up Open Up
Group Num: 8
LCP           State: OPENED
IPCP          State: OPENED
Multilink     State: OPENED
Multi-Class   State: CLOSED

Bundle Size:             1
Class Level Tx:          1
            Rx:          1
Max Load Threshold:      0
Bundle Tx Load Avg:      0
Bundle Rx Load Avg:      0
No Of Pck  in Rx Buf Q: 0
Lowest link Speed:       1984000
Max Fragment Size:       256
High Pri Member link is Serial 1/0:0

Rx Stats
 Total:              0
   Data:             0
   Control:          0
   Null:             0
 Discard:
   Pck Too Long:     0
```

```
   Invalid Proto:  0
   Wrong Proto:    0
   Padding Error:  0
   Invalid Cls#:   0
   Error to CP:    0
   No Lower Lyr:   0
   No Upper Lyr:   0
   Others:         0

Tx Stats
 Total:            0
   Data:           0
   Control:        0
   Null:           0
 Discard:
   Pck Too Long:   0
   No Lower Lyr:   0
   EnQueue Full:   0
   Others:         0
```

The following is is sample output *with* Multi-Class configured:

```
********** MLPPP Bundle Stats **********
Multilink 8: MLPPP is Admin Up / Oper Up
Group Num: 8
LCP         State: OPENED
IPCP        State: OPENED
Multilink   State: OPENED
Multi-Class State: OPENED

Multilink header format is LONG SEQ NUM
Class suspendable level is 5 tx classes and 5 rcv classes

Max Fragment delay is 10 ms
Bundle Size:            1
Class Level Tx:         5
            Rx:         5
Max Load Threshold:     0
Bundle Tx Load Avg:     0
Bundle Rx Load Avg:     0
No Of Pck  in Rx Buf Q: 0
Lowest link Speed:      1984000
Max Fragment Size:      256
High Pri Member link is Serial 1/0:0

Rx Stats
 Total:            0
   Data:           0
   Control:        0
   Null:           0
 Discard:
```

```
   Pck Too Long:    0
   Invalid Proto:   0
   Wrong Proto:     0
   Padding Error:   0
   Invalid Cls#:    0
   Error to CP:     0
   No Lower Lyr:    0
   No Upper Lyr:    0
   Others:          0

Tx Stats
 Total:             0
   Data:            0
   Control:         0
   Null:            0
 Discard:
   Pck Too Long:    0
   No Lower Lyr:    0
   EnQueue Full:    0
   Others:          0
```

Refer to the **show interface multilink** command page 122 for parameter descriptions.

show ppp interface multilink/dialer multi-class

This command displays Multi-Class MLPPP status and statistics.

Syntax

show ppp interface [*type* | *type number*] **multi-class**

type	Multilink or Dialer interfaces upon which PPP is running.
number	Designation for either Multilink or Dialer interfaces.

Mode

EXEC: **XSR>**

Sample Output

The following example displays output of this command:

```
********** MLPPP Bundle MultiClass Stats **********
Multilink 1: MLPPP is Admin Up / Oper Up
Group Num: 1
LCP           State: OPENED
IPCP          State: OPENED
Multilink     State: OPENED
Multi-Class   State: OPENED

Multilink header format is LONG SEQ NUM
Class suspendable level is 5 tx classes and 5 rcv classes
```

```
Max Fragment delay is 10 ms
Max Fragment Size is 256 bytes

Class                 0          1          2          3          4
 QoSCls#             -1          0          1          2          3
 ExpctSeq#            1          1          1          1          1
 LastFwdSeq#          0          0          0          0          0
 LastM#               0          0          0          0          0
 maxFListSize         0          0          0          0          0
 FragListSize         0          0          0          0          0
 TxSeq#               1          1          1          1          1
 TxBufferSize         0          0          0          0          0

 Rx Load
  Average             0          0          0          0          0
  Max                 0          0          0          0          0
  Min                 0          0          0          0          0
 Tx Load
  Average             0          0          0          0          0
  Max                 0          0          0          0          0
  Min                 0          0          0          0          0

 Rx Stats:
  Total               0          0          0          0          0
  Discard
    SeqError          0          0          0          0          0
    FListFull         0          0          0          0          0
    Seq<Exp           0          0          0          0          0
    NoBgnFlg          0          0          0          0          0
    AddFgFail         0          0          0          0          0
    CleanQ            0          0          0          0          0
 Tx Stats:
  Total               0          0          0          0          0
  Discard
    CleanQ            0          0          0          0          0
    QFull             0          0          0          0          0
```

PPP Multilink Multi-Class Bundle Parameter Descriptions

Class

Range	0 - 4
Description	Suspendable class level - 0: default class lowest level: 4: highest level.

QoSCls#

Range	-1 - 3

Description	Equivalent QoS class, • -1: fair class. • 0: low priority class. • 1: normal priority class. • 2: medium priority class. • 3: high priority class.

ExpctSeq#

Range	-1 - 16777215
Description	Next expected sequence number of receiving fragment for this class.

LastFwdSeq#

Range	-1 - 16777215
Description	Last forwarded sequence number of the fragment of this class to the upper layer.

LastM#

Range	-1 - 16777215
Description	Last M (the smallest received sequence number) of all the member links in this class to the upper layer.

MaxFListSize

Range	Not defined.
Description	Maximum receive fragment reassemble list size for this class. Reset when a **`show`** command is issued.

FragListSize

Range	Not defined.
Description	Current receive fragment reassemble list size for this class.

TxSeq#

Range	-1 - 16777215
Description	Last sequence number transmitted in this class.

TxBufferSize

Range	0-1
Description	Current transmit buffer size for this class.

Tx/Rx Load

Average/Max/Min

Range	0-255
Description	Transmit/receive load for this class against the total bandwidth, 255=100%

Rx Stats

Total	Sum of fragments received for this class.
Discard Seq Error	Sum of received fragments discarded for this class because sequence number is out of order.
FlistFull	Sum of received fragments discarded for this class because fragment list is full.
Seq<Exp	Sum of received fragment discarded for this class because sequence number is less than expected.
NoBgnFlg	Sum of received fragments discarded for this class because no BEGIN flag detected.
AddFgFail	Sum of received fragments discarded for this class because fragment cannot be added into fragment list.
CleanQ	Sum of received fragments discarded for this class while cleaning the interface.

Tx Stats

Total	Sum of fragments transmitted for this class.
Discard CleanQ	Sum of transmission fragments discarded for this class while cleaning port.
Qfull	Sum of transmission fragments discarded for this class because transmission queue is full.

show ppp interface multilink/dialer memberlink

This command displays general member link statistics under MLPPP or specific member link statistics if specified.

Syntax

show ppp interface multilink *<1-32767>* **memberlink** [*type number*]
show ppp interface dialer *<1-256>* **memberlink** [*type number*]

Parameters

type	Interface type serial. If serial is specified, only this serial member link statistics display, otherwise all member link data display.
number	Card/port numbers for a serial interface. Card/port:channel numbers for serial channel groups.

Mode

EXEC: **XSR>**

Sample Output

The following example displays output of this command:

```
********** MLPPP Member Link Stats **********
Multilink 1: MLPPP is Admin Up / Oper Up
Group Num: 1
LCP              State: OPENED
```

```
IPCP              State: OPENED
Multilink         State: OPENED
Multi-Class       State: OPENED

Multilink header format is LONG SEQ NUM
Class suspendable level is 5 tx classes and 5 rcv classes

Serial 1/0:0
 Tx: Total           0   Discard          0(0/0)
 Rx: Total           0   Discard          0
```

PPP Multilink Member Link Paremeter Descriptions

The detail of transmit/receive statistics for the member link

Serial 1/00	Name of the member link.

Tx

Total	Sum of fragments transmitted over this member link.
Discard	Sum of transmitting fragments discarded over this member due to invalid length or no lower layer.

Rx

Total	Sum of fragments received over this member link.
Discard	Sum of received fragments discarded over this member link.

show ppp interface multilink/dialer memberlink multi-class

This command displays multi-class statistics on the member link under MLPP.

Syntax

show ppp interface multilink <*1-32767*> **memberlink multi-class** <*type number*>
show ppp interface dialer <*1-256*> **memberlink multi-class** <*type number*>

Parameters

type	Interface type Serial. If serial is specified, only this serial member link statistics display, otherwise all member link data display.
number	Card/port numbers for a Serial port. Card/port:channel numbers for Serial channel groups.

Mode

EXEC: **XSR>**

Sample Output

The following example displays output of this command:

```
********** MLPPP Member Link MultiClassStats **********
Multilink 1: MLPPP is Admin Up / Oper Up
Group Num: 1
LCP           State: OPENED
IPCP          State: OPENED
Multilink     State: OPENED
Multi-Class   State: OPENED

Multilink header format is LONG SEQ NUM
Class suspendable level is 5 tx classes and 5 rcv classes

Class                 0          1          2          3          4
Serial 1/0:0
 LastRxSeq#           0          0          0          0          0
 LastTxSeq#           0          0          0          0          0
 Rx Stats:
  Total               0          0          0          0          0
  Discard
   FListFull          0          0          0          0          0
   Seq#Err            0          0          0          0          0
   Seq<Expt           0          0          0          0          0
   NoBegin            0          0          0          0          0
   AddFrgFail         0          0          0          0          0
   CleanQ             0          0          0          0          0
Tx Stats:
  Total               0          0          0          0          0
  Discard
   CleanQ             0          0          0          0          0
   QFull              0          0          0          0          0
```

PPP Multilink Member Link Multi-Class Parameter Descriptions

Class

Range	0 - 4
Description	Level of suspendable class, 0 default class lowest suspendable level 4 the highest suspendable level
Serial 1/00	Name of the member link.

LastRXSeq#

Range	-1 - 16777215
Description	Last sequence number of fragment sent over the member link for this class.

LastRXSeq#

Range	-1 - 16777215
Description	Last sequence number of fragment received over the member link for this class.

Rx Stats

Total	Sum of fragments received for this class.
Discard SeqError	Sum of received fragments discarded for this class because sequence number is out of order over this member link.
FlistFull	Sum of received fragments discarded for this class over this member link because fragment list is full.
Seq<Exp	Sum of received fragments discarded for this class over this member link because sequence number is less than expected.
NoBgnFlg	Sum of received fragments discarded for this class over this member link because no BEGIN flag is deteced.
AddFgFail	Sum of received fragments discarded for this class over this member link because fragment can not be added to the fragment list.
CleanQ	Sum of received fragments discarded for this class over this member link while cleaning the interface.

Tx Stats

Total	Sum of fragments transmitted for this class under this member link.
Discard CleanQ	Sum of transmission fragments discarded for this class under this member link during interface cleaning.
Qfull	Sum of transmission fragments discarded for this class under this member link because transmission queue is full.

show ppp interface dialer x mlpppgroup x bap

This command displays BAP multilink bundle statistics of a specific bundle under the dialer interface. You can view individual multilink bundles when more than one exists on the dialer interface.

Syntax

show ppp interface dialer <*number*> **mlpppgroup** <*number*> **bap**

number	Dialer interface number, ranging from 0 to 255.
number	Multilink bundle number, ranging from 0 to 255.

Mode

EXEC: **XSR>**

Sample Output

The following is sample output from the command:

```
********** MLPPP Bundle Stats **********
Dialer1: MLPPP is Admin Up / Oper Up
Group Num: 1
LCP          State: OPENED
IPCP         State: OPENED
```

```
BACP        State: OPENED
Multilink   State: OPENED
Multi-Class State: OPENED

Multilink header format is LONG SEQ NUM
Class suspendable level is 5 tx classes and 5

Max Fragment delay is 10 ms
Bundle Size:            20
Class Level Tx:         5
            Rx:         5
Max Load Threshold:     100
Bundle Tx Load Avg:     0
Bundle Rx Load Avg:     0
No Of Pck  in Rx Buf Q: 0
Lowest link Speed:      64000
Max Fragment Size:      64
High Pri Member link is Serial 3/2/0:10

Rx Stats
 Total:            20137
   Data:           19103
   Control:        2
   Null:           1032
 Discard:
   Pck Too Long:   0
   Invalid Proto:  0
   Wrong Proto:    0
   Padding Error:  0
   Invalid Cls#:   0
   Error to CP:    0
   No Lower Lyr:   0
   No Upper Lyr:   0
   Others:         18

Tx Stats
 Total:            10891
   Data:           9799
   Control:        42
   Null:           1050
 Discard:
   Pck Too Long:   0
   No Lower Lyr:   0
   EnQueue Full:   0
   Others:         0

BAP information:
Local has precedence

Rcv Call-Req:         0
```

```
Rcv Call-ReqAck:       19
Rcv CallBack-Req:      0
Rcv CallBack-ReqAck:   0
Rcv LinkDrop-Req:      0
Rcv LinkDrop-ReqAck:   0
Tx Call-Req:           20
Tx Call-ReqAck:        0
Tx CallBack-Req:       0
Tx CallBack-ReqAck:    0
Tx LinkDrop-Req:       0
Tx LinkDrop-ReqAck:    0

Discriminators        Local    Remote
Serial 3/2/0:26           0         1
Serial 3/2/0:30           1         3
Serial 3/2/0:29           2         5
Serial 3/2/0:28           3         7
Serial 3/2/0:27           4         9
Serial 3/2/0:25           5        11
Serial 3/2/0:24           6        13
Serial 3/2/0:23           7        15
Serial 3/2/0:22           8        17
Serial 3/2/0:21           9        19
Serial 3/2/0:20          10        21
Serial 3/2/0:14          11        23
Serial 3/2/0:19          12        25
Serial 3/2/0:18          13        27
```

9

Configuring Frame Relay

Observing Syntax and Conventions

CLI command syntax and conventions use the notation described below.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[*x*]	[ ] Square brackets indicate an optional parameter (*italic*)
[*x* | *y* | *z*]	[|] Square brackets with vertical bar indicate a choice of values
{*x* | *y* | *z*}	{ | } Braces with vertical bar indicate a choice of a required value
[*x* {*y* | *z*}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<*xx*>)	*xx* signifies the interface type, class map, policy map or other value you specify; e.g., **`F1`**, **`G3`**, **`M57`**, **`S2/1.0`**, **`Node Name`**., **`DLCI class name`**
Next Mode entries display the CLI prompt after a command is entered.	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

Frame Relay Commands

This chapter describes the configurable features of the Frame Relay interface for the XSR in the following command subsets:

- "Frame Relay Map Class Commands" on page 9-95
- "Frame Relay Clear and Show Commands" on page 9-102

encapsulation frame-relay

This command enables Frame Relay encapsulation on an interface using IETF (RFC-2427) encapsulation format. When connecting to non-XSR servers, be sure the remote end is configured for IETF encapsulation unless the remote end can handle IETF-formatted Frame Relay headers. Other routers may be configured using the following command:

```
encapsulation frame-relay IETF
```

Note: If encapsulation is changed from one type to another, all related values of the current encapsulation and any sub-interface settings are deleted. Also, once encapsulation is set on an interface, any sub-interface of that port created later is automatically encapsulated. Finally, you must first enter the **`no encapsulation`** command to change the encapsulation type.

Syntax

encapsulation frame-relay

Syntax of the "no" Form

Disable Frame Relay encapsulation on the interface with the *no* form:

no encapsulation frame-relay

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

This example sets Frame Relay encapsulation on interface serial 1/0:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation frame-relay
XSR(config-if<S1/0>)#no shutdown
```

frame-relay class

This command associates a map class to an interface or sub-interface. It can be applied to both Frame Relay interfaces and sub-interfaces.

Note: Frame Relay *traffic shaping must be enabled* on the interface for this command to be effective.

Each virtual circuit (DLCI) created on the interface or sub-interface inherits all relevant parameters defined in the named map class. For each virtual circuit, the precedence rules are as follows:

- Use the map class associated with the virtual circuit if it is configured:

  frame-relay interface-dlci *dlci-num*

  class *map-class-name*

- If not, use the map class associated with the sub-interface if the map class exists:

  interface serial *1/0.1 point*

  frame-relay *class sub-interface-map-class-name*

- If not, use the map class associated with the interface if the map class exists:

  interface serial *1/0*

  frame-relay class *interface-map-class-name*

- If not, use the interface default parameters (CIR: 56 kbps, Bc and Be: 7000 bits, adaptive shaping: disabled and service-policy: not set).

Syntax

frame-relay class *name*

name	Name of the map class.

Syntax of the "no" Form

The *no* form removes the association of the map class to the interface or sub-interface:

no frame-relay class *name*

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

The following commands set Frame Relay map classes *fastlink* and *normlink* with an outbound CIR value of *56* kbps and *25.6* kbps, respectively:

```
XSR(config)#map-class frame-relay fastlink
XSR(config-map-class<fastlink>)#frame-relay cir out 56000
XSR(config)#map-class frame-relay normlink
XSR(config-map-class<normlink>)#frame-relay cir out 25600
```

The following commands direct serial link *1/0* to use QoS values from the *normlink* map class unless explicitly overridden.

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation frame-relay
XSR(config-map-class<fastlink>)#frame-relay traffic-shaping
XSR(config-if<S1/0>)#no shutdown
XSR(config-if<S1/0>)#frame-relay class normlink
```

The following commands configure sub-interface *serial 1/0.2* to use a different map class (*fastlink*) than that specified for *serial 1/0*.

```
XSR(config)#interface serial 1/0.2 point-to-point
XSR(config-subif<S1/0.2>)#no shutdown
XSR(config-subif<S1/0.2>)#frame-relay class fastlink
```

frame-relay interface-dlci

This command assigns a data-link connection identifier (DLCI) to a specified Frame Relay sub-interface. It is used for sub-interfaces only. When you invoke this command, you enter Frame Relay DLCI Interface mode. This provides the following command options, which must be used with the relevant class names you previously assigned:

- **class** *name* - assigns a map class to a DLCI.
- **no class** *name* - cancels the relevant class.
- **exit** - quits Frame Relay DLCI *interface* mode.

If you attempt to create a DLCI which has already been configured, the following sample warning will be issued:

```
DLCI 43 is already configured on sub-interface 3
```

Note: You must delete an existing DLCI before the same DLCI can be created on a different sub-interface of the Frame Relay interface.

Once chosen as static, no inverse ARP will be sent out by default. A free inverse ARP request (similar to above) can be requested by this command.

Once chosen as static, this DLCI can be made to respond to a broadcast bootp message entering on this DLCI from the frame-relay network. Non-broadcast bootp will still be sent to the local DHCP server or relayed to the IP helper address server..

Notes: The remote site must support sending inverse-arp responses or the interface will come down.

An inverse arp is sent from the XSR at a rate of 1 every 4 seconds. It is not configurable.

Syntax

frame-relay interface-dlci nn [[*keep-alive nn* [*gratuitous-inverse-arp*]] | [*gratuitous-inverse-arp* [*keep-alive nn*]] | [*ip A.B.C.D* [[*bootp* [[*gratuitous-inverse-arp* [*keep-alive nn*]] | [*keep-alive nn* [*gratuitous-inverse-arp*]]]] | [*gratuitous-inverse-arp* [[*bootp* [*keep-alive nn*]] | [*keep-alive nn* [*bootp*]]]] | [*keep-alive nn* [[*gratuitous-inverse-arp* [*bootp*]] | [*bootp* [*gratuitous-inverse-arp*]]]]]]]

interface-dlci nn	DLCI number for the sub-interface, ranging from 16 to 1007. For the Point-to-Point (P2P) sub-interface type, only one DLCI is allowed. For Point-to-Multi-Point (P2MP) you can configure multiple DLCIs.
gratuitous-inverse-arp	Sends inverse ARP request and ignores a response. This parameter occurs for *non-static* IP mapping. P2P sub-interfaces will generate a free inverse arp to allow the remote side to learn the IP address of this sub-interface. This parameter is useful only for P2P sub-interfaces, since Point-to-MultiPoint interfaces withdynamic IP resolution will always inverse ARP to learnthe remote node's IP address. Omitting this value in a P2P sub-interface prevents sending an inverse-arp request. An inverse-arp responseis always sent when an inverse-arp request is received. Broadcast bootp is not supported in dynamic mode. Allbootp request in this mode are forwarded.
ip	Protocol type to set static IP address to DLCI mapping.
A.B.C.D	Static IP address of peer node. No address checking done.
bootp	Respond to a broadcast bootp request with static IP `address` (used for Remote Auto Install Central Site).
gratuitous-inverse-arp	Sends inverse ARP request. Response is ignored. Valid `for` both MP2P & P2P sub-interfaces.
keep-alive nn	*nn* refers to the duration that a DLCI under a P2P interface will wait with no traffic being received before sending an inverse-arp packet to confirm that the remote side is still present. The nn range is 10 to 600 seconds.

Syntax of the “no” Form

Use the *no* command to delete the DLCI from the specified sub-interface:

no frame-relay interface-dlci *dlci-num*

Mode

Sub-interface configuration: **XSR(config-subif<*xx*>)#**

Next Mode

Frame Relay DLCI configuration: **XSR(config-fr-dlci<xx>)#**

Examples

The following example maps DLCIs *16* and *18* on serial sub-interface *1/0.1* to the specified IP addresses, supporting bootp and sending a free inverse ARP. Also, DLCI *17* is configured on sub-interface *1/0.2*, a free inverse ARP is sent, and emote keep-alive is supported in P2P mode.

```
XSR(config)#interface serial 1/0.1 multi-point
XSR(config-subif)#ip helper 10.10.1.2
XSR(config-subif)#ip address 133.133.1.1 255.255.255.0
XSR(config-subif)#frame-relay interface-dlci 16 ip 133.133.1.2 gratuitous-
inverse-arp bootp
XSR(config-fr-dlci)#frame-relay interface-dlci 18 ip 133.133.1.3 bootp
XSR(config-fr-dlci)#no shutdown
XSR(config-fr-dlci)#interface serial 1/0.2 point-to-point
XSR(config-subif)#ip helper 10.10.1.2
XSR(config-subif)#ip address 133.134.1.1 255.255.255.0
XSR(config-subif)#frame-relay interface-dlci 17 gratuitous-inverse-arp keep-alive
30
XSR(config-fr-dlci)#no shutdown
```

frame-relay intf-type

This command defines the Frame Relay interface type for the interface. The XSR works as a UNI device only, with DTE or DCE as valid entries.

Syntax

frame-relay intf-type {*dte* | *dce*}

dte	Specifies the XSR to act as a Frame Relay DTE UNI device.
dce	Specifies the XSR to act as a Frame Relay DCE UNI device.

Syntax of the “no” Form

no frame-relay intf-type {*dte* | *dce*}

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Default

dte

Examples

The following example configures Serial interface *1/0* to act as a Frame Relay *DTE*, and to use the ANSI Annex-D LMI:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#no shutdown
XSR(config-if<S1/0>)#encapsulation frame-relay
XSR(config-if<S1/0>)#frame-relay intf-type dte
XSR(config-if<S1/0>)#frame-relay lmi-type ansi
```

The following example configures Serial interface *1/0* to act as a Frame Relay *DCE*, and to use the ANSI Annex-D LMI:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#no shutdown
XSR(config-if<S1/0>)#encapsulation frame-relay
XSR(config-if<S1/0>)#frame-relay intf-type dce
XSR(config-if<S1/0>)#frame-relay lmi-type ansi
```

frame-relay lmi-t391dte

This command sets the interval between LMI Link Integrity Verification (LIV) message transmissions on the Data Terminal Equipment (DTE) interface.

Note: On third-party devices, the LMI LIV period may be configured using the KeepAlive configuration on the interface.

Syntax

frame-relay lmi-t391dte *period_in_sec*

period_ in_sec	Sets the interval between LMI LIV polls, ranging from 5 to 330 seconds.

Syntax of the “no” Form

Use the *no* command to restore the default interval value:

no frame-relay lmi-t391dte

Default

10

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

Refer to the example in the **lmi-n391dte** command on page 89.

frame-relay lmi-n391dte

This command sets the full status-polling interval when the Digital Terminal Equipment (DTE) interface is configured to set the full status message-polling interval.

Syntax

frame-relay lmi-n391dte *num_ka-exchanges*

num_ka-exchanges	Number of keep-alive exchanges to occur before requesting a full status message, ranging from 1 to 255.

Syntax of the "no" Form

The *no* form of this command restores the default interval value:

no frame-relay lmi-n391dte

Default

6

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

This example establishes that a status inquiry will be sent every *five* seconds and that one of every *ten* status inquiries generated will request a full status response from the Frame Relay switch. The other nine status inquiries will request keep-alive exchanges only:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation frame-relay
XSR(config-if<S1/0>)#frame-relay intf-type dte
XSR(config-if<S1/0>)#frame-relay lmi-n391dte 10
XSR(config-if<S1/0>)#frame-relay lmi-t391dte 5
XSR(config-if<S1/0>)#no shutdown
```

frame-relay lmi-n392dce

This command sets the error threshold on a Data Communications Equipment (DCE) interface.

Syntax

frame-relay lmi-n392dce *threshold*

threshold	Error threshold, ranging from 1 to 10.

Syntax of the "no" Form

The *no* form of this command removes the current setting:

no frame-relay lmi-n392dce

Default

3

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

This example sets the LMI failure threshold to *5* for the *DCE* device:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation frame-relay
XSR(config-if<S1/0>)#frame-relay intf-type dce
XSR(config-if<S1/0>)#frame-relay lmi-n392dce 5
```

frame-relay lmi-n392dte

This command sets the error threshold on a Data Terminal Equipment (DTE) interface.

Syntax

frame-relay lmi-n392dte *threshold*

threshold	Error threshold, ranging from 1 to 10.

Syntax of the “no” Form

Use the *no* command to remove the current setting:

```
no frame-relay lmi-n392dte
```

Default

3

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example sets the LMI failure threshold to *5* for the *DTE* device:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation frame-relay
XSR(config-if<S1/0>)#frame-relay intf-type dte
XSR(config-if<S1/0>)#frame-relay lmi-n392dte 5
```

frame-relay lmi-t392dce

This command sets polling verification timer on a Data Communications Equipment (DCE) interface. The timer marks the duration that the DCE expects to receive a Status Enquiry from a DTE device.

Syntax

frame-relay lmi-t392dce *period_in_sec*

events	Interval to wait for a Status Enquiry, ranging from 5 to 30 seconds.

Syntax of the "no" Form

The *no* form of this command restores the default interval:

no frame-relay lmi-t392dce

Default

15 seconds

Example

The following example sets the DCE to wait *20* seconds for a status enquiry from the DTE before declaring an error event:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation frame-relay
XSR(config-if<S1/0>)#frame-relay intf-type dce
XSR(config-if<S1/0>)#frame-relay lmi-t392dce 20
```

frame-relay lmi-n392dce

This command sets the error threshold on a Data Communications Equipment (DCE) interface.

Syntax

frame-relay lmi-n392dce *threshold*

threshold	Error threshold, ranging from 1 to 10.

Syntax of the "no" Form

The *no* form of this command removes the current setting:

no frame-relay lmi-n392dce

Default

3

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

This example sets the LMI failure threshold to *5* for the *DCE* device:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation frame-relay
XSR(config-if<S1/0>)#frame-relay intf-type dce
XSR(config-if<S1/0>)#frame-relay lmi-n392dce 5
```

frame-relay lmi-n393dce

This command sets the monitored event count on a Data Communications Equipment (DCE) interface.

Syntax

frame-relay lmi-n393dce *events*

events	Value of monitored events count ranging from 1 to 10.

Syntax of the “no” Form

The *no* form of this command removes the current setting:

```
no frame-relay lmi-n393dce
```

Default

4

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

This example sets the LMI monitored events count to *10* on serial port *1/0*:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation frame-relay
XSR(config-if<S1/0>)#frame-relay lmi-n393dce 10
```

frame-relay lmi-type

This command configures the Local Management Interface (LMI) type on a per-interface basis.

Syntax

frame-relay lmi-type {*ilmi* | *ansi* | *q933a* | *auto* | *none*}

ilmi	Interim LMI (FRF 1.1).
ansi	Annex D defined by American National Standards Institute (ANSI) standard T1.617.
q933a	ITU-T Q.933 Annex A.
auto	The port will attempt to detect and match the LMI type used by the attached Frame Relay switch.
none	No LMI used. This is meant to test or connect XSRs directly.

Syntax of the "no" Form

Use the *no* command to return to the default LMI type:

no frame-relay lmi-type {*ilmi* | *ansi* | *q933a* | *auto* | *none*}

Default

auto

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

This example sets serial interface *1/0* to use the *ANSI Annex-D* LMI:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation frame-relay
XSR(config-if<S1/0>)#frame-relay lmi-type ansi
XSR(config-if<S1/0>)#no shutdown
```

frame-relay traffic-shaping

This command enables map-class parameters for all Permanent Virtual Circuits (PVCs) on a Frame Relay port. For virtual circuits which have no specific traffic shaping or queuing parameters specified, a set of default values is used.

Syntax

frame-relay traffic-shaping

Syntax of the "no" Form

The *no* command disables the use of map-class parameters:

```
no frame-relay traffic-shaping
```

Default

Disable

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

This example enables both traffic shaping and per-virtual circuit queuing:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation frame-relay
XSR(config-if<S1/0>)#frame-relay traffic-shaping
XSR(config-if<S1/0>)#no shutdown
```

interface

This command selects a physical port for configuration as a router interface. The XSR supports FastEthernet or GigabitEthernet, serial, and T1/E1/ISDN-PRI physical ports. For configuration purposes, all serial ports and T1/E1/ISDN-PRI channel groups are treated as a serial port.

Optionally, you can set up the Console port as a *WAN* interface for dial backup purposes (refer to the *Caution* below).

Caution: Be aware that when you enable the Console port as a WAN port, you can no longer directly connect to it because it is in data communication mode. Your only access to the CLI will be to Telnet to an IP address of a configured port. Also, if your *startup-config* file does not configure any ports properly and sets up the console port as a serial interface, you will no longer be able to login and will have to press the Default button to erase your configuration.

Syntax

interface serial *port_num interface_num*

port_num interface _num	The physical port and interface number. An interface number for a serial interface can be comprised of: *card_num/NIM_num/port_within_NIM*. For example, *0/1/2* sets physical port 2 on the NIM card in slot 1 of the motherboard. Leading zeros in *interface_num* can be omitted. So *0/1/2* is the same as *1/2*. If the serial port resides on a T1/E1 port, then channel group data must be added at the end of the string to mark which channel group of the T1/E1 port will be set: *card_num/NIM_num/ port_within_NIM*: *[channel-group_num].* For example, *0/2/ 1:15* sets channel-group *15* of the T1 or E1 port 1 in NIM slot *2* on the motherboard.

Note: Leading zeros defined in *interface_num* can be omitted. For example, *0/1/2* is equivalent to *1/2.*

Syntax of the "no" Form

The *no* command deletes the interface:

no interface serial *port_num interface_num*

Note: You cannot directly delete a Serial interface assigned to a T1/E1 channel group. You must instead delete a channel group to erase the Serial port.

Mode

Global configuration: **XSR(config)#**

Examples

This example selects interface serial *1/0* and sets Frame Relay encapsulation:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation frame-relay
XSR(config-if<S1/0>)#no shutdown
```

The following example selects channel group *12* of the T1/E1 port1 on the second NIM card so that later configurations will apply to this serial port:

```
XSR(config)#interface serial 2/1:12
XSR(config-if<s2/1:12)#encapsulation frame-relay
XSR(config-if<S1/0>)#no shutdown
```

Frame Relay Map Class Commands

class

This command assigns a map class to a specific Data-Link Connection Identifier (DLCI). This can be used to override the default values for the DLCIs or to override a class assigned to the interface or sub-interface that the DLCI belongs to.

The actual map class is defined using the **map-class frame-relay** command in Global configuration mode. This command only applies to assigning a map class to DLCIs.

Syntax

class *name*

name	Name of the map class to associate with this DLCI, up to 29 characters.

Syntax of the "no" Form

The *no* command removes the assigned map class from the DLCI.

no class *name*

Mode

Virtual Circuit configuration: **`XSR(config-fr-dlci)#`**

Example

The first three commands in the following example set up Serial sub-interface *1/0.1* with associated DLCI *16*. The last two commands define map class *Hello*.

```
XSR(config)#interface serial 1/0.1 point-to-point
XSR(config-if<S1/0>)#interface serial 1/0.1 point-to-point
XSR(config-subif)#frame-relay interface-dlci 16
XSR(config-fr-dlci)#class Hello
XSR(config)#map-class frame-relay Hello
XSR(config-map-class<Hello>)#frame-relay cir out 128000
```

frame-relay adaptive-shaping

This command enables and selects the mechanism to trigger adaptive shaping, the dynamic imposition of traffic shaping parameters (CIR, Bc, Be) based on external feedback indicating upstream congestion conditions.

Frame Relay switches use BECN (Back End Congestion Notification) to indicate congestion and throttle the DTE traffic rate.

Syntax

```
frame-relay adaptive-shaping
```

Syntax of the “no” Form

The *no* command disables adaptive shaping:

```
no frame-relay adaptive-shaping
```

Mode

Map Class configuration: **`XSR(config-map-class)#`**

Default

Disabled

Example

This example sets Frame Relay map-class *normlink* with traffic shaping:

```
XSR(config)#map-class frame-relay normlink
XSR(config-map-class)#frame-relay adaptive-shaping
```

frame-relay bc

This command specifies the outgoing Committed burst size (Bc) for a Frame Relay map-class. Committed burst is specified in bits, but an implicit time factor is derived from the sampling interval (Tc) on the switch, which is defined as the burst size divided by the Committed Information Rate (CIR). This is expressed in the formula: *Tc = Bc/CIR*. For more information, refer to “frame-relay cir” on page 98.

Syntax

frame-relay bc out *bits*

out	Sets the traffic direction - *output* rate limiting only.
bits	Committed burst size, in bits.

Syntax of the "no" Form

The *no* command resets the committed burst size to its default value:

no frame-relay bc out

Mode

Map Class configuration: **XSR(config-map-class)#**

Default

7000 bits

Example

This example creates the map class *slowlink* with *bc* set to *6000* bits:

```
XSR(config)#map-class frame-relay slowlink
XSR(config-map-class<slowlink>)#frame-relay bc out 6000
```

frame-relay be

This command specifies the outgoing excess Burst size (Be) for a Frame Relay map-class.

Syntax

frame-relay be out *bits*

out	Sets the traffic direction - *output* rate limiting only.
bits	Committed burst size in bits.

Syntax of the "no" Form

The *no* command resets the committed burst size to its default value:

no frame-relay be out

Mode

Map Class configuration: **XSR(config-map-class)#**

Default

7000 bits

Example

This example adds map class *slowlink* with *Be* of *10000* and *Bc* of *6000* bits:

```
XSR(config)#map-class frame-relay slowlink
XSR(config-map-class<slowlink>)#frame-relay be out 10000
XSR(config-map-class<slowlink>)#frame-relay bc out 6000
```

frame-relay cir

This command specifies the outgoing Committed Information Rate (CIR) for a Frame Relay map-class. CIR, Bc and Be values specify how the XSR forwards packets under normal and congested conditions using the following equation:

Tc = Bc/CIR = 7,000 bits / 56,000 bps = 125 mS (Bc and CIR values are default)

Frame Relay networks are committed to deliver *Bc* bits of data every *Tc*, so maximum committed throughput equals 7,000/125mS = 56kbps = CIR. In this sense, Committed Burst (Bc) is not really a burst but a "smoothing" function for the number of bits that the XSR is allowed to transmit during the Tc period in order to achieve the specified CIR.

Since the maximum number of bits that can be sent during Tc is Bc plus Be bits, using the default values, maximum throughput equals (Bc + Be)/Tc = (7,000 + 7,000)/125mS = 112kbps = 2 * 56kbps = 2 * CIR.

Syntax

frame-relay cir out *rate*

out	Sets the traffic direction - *output* rate limiting only.
rate	CIR, ranging from 1000 to 1,000,000 bits per second.

Syntax of the "no" Form

The *no* command resets the CIR to its default value:

```
no frame-relay cir out
```

Mode

Map Class configuration: **XSR(config-map-class)#**

Defaults

- CIR enforced for outgoing traffic only
- CIR: 56000 bps
- Be: 7000 bits
- Bc: 7000 bits

Example

This example creates the map class *slowlink* with *cir* set at *9600* bps:

```
XSR(config)#map-class frame-relay slowlink
XSR(config-map-class<slowlink>)#frame-relay cir out 9600
```

frame-relay fragment

This command specifies the FRF.12 end-to-end fragment size for a Frame Relay map-class. Fragment size is defined in bytes. It specifies the number of payload bytes from the original frame that will go into each fragment. The transmitted fragment will include eight additional bytes from headers (6) and CRC(2).

Note: For proper operation of fragmentation, QOS is required to classify a service-policy which will define a high priority queue. The queue must send frames no larger than the fragment size or fragmentation will also be applied to high priority queue data and latency will grow, defeating the primary purpose of FRF.12 fragmentation.

Syntax

frame-relay fragment *bytes*

bytes	Size of frame to pass unfragmented.

Syntax of the "no" Form

The *no* command disables FRF.12 end-to-end fragmentation:

no frame-relay fragment

Mode

Map Class configuration: **XSR(config-map-class)#**

Default

Fragmentation is disabled

Example

The following example creates the map class *slowlink* with fragmentation set at *53* bytes:

```
XSR(config)#map-class frame-relay slowlink
XSR(config-map-class<slowlink>)#frame-relay fragment 53
XSR(config-map-class<slowlink>)#service-policy frf12
```

map-class frame-relay

The command selects a supported Frame Relay map class and gives it a mnemonic name that can be referenced in Frame Relay configuration.

Map-class frame-relay starts configuration of a map-class profile with a user-specific name. When a map-class command is entered, the CLI enters Map-Class configuration mode, changing the CLI prompt to **config-map-class** where you can enter map-class specific values.

Syntax

map-class [**frame-relay** | **dialer**] *map-class-name*

frame-relay	Sets a Frame Relay map class.

dialer	Sets a dialer map class. For more information, refer to "Configuring the Dialer Interface" on page 83.
map-class-name	Name of the map class to associate with this DLCI, up to 29 characters.

Syntax of the "no" Form

no map-class [**frame-relay** | **dialer**] *map-class-name*

Mode

Global configuration: **XSR(config)#**

Next Mode

FR Map-Class configuration: **XSR(config-map-class)#**

Example

This example defines frame relay map-class *normlink*:

```
XSR(config)#map-class frame-relay normlink
XSR(config-map-class<normlink>)#frame-relay adaptive-shaping
XSR(config-map-class<normlink>)#frame-relay cir out 64000
XSR(config-map-class<normlink>)#frame-relay bc out 8000
XSR(config-map-class<normlink>)#frame-relay be out 8000
XSR(config-map-class<normlink>)#service-policy output HighPriority
```

service-policy

This command sets the service-policy profile for the class map. The service-policy is a flexible method to configure QoS for an interface, sub-interface and DLCI, You can use it to create priority queues, custom queues, WFQ or FIFO queues. Refer to "Configuring Quality of Service" on page 83 for more details.

Syntax

service-policy {**out**} **service-policy-name**

out	Service policy applies to outgoing traffic only.
service-policy-name	Name of the separated configured service-policy profile to apply for this map-class.

Syntax of the "no" Form

The *no* form of this command disables a service-policy:

no service-policy output service-policy-name

Mode

Map Class configuration: **XSR(config-map-class)#**

Example

The following example specifies *HighPriority* as the policy for the class map:

```
XSR(config-map-class)#service-policy out HighPriority
```

shutdown

This command disables an interface or sub-interface. A sub-interface is shut down (no longer passing data) when one of the following occurs:

- An explicit **shutdown** command is entered on the sub-interface.
- A **shutdown** command is issued on the parent Frame Relay interface of this sub-interface.
- A **shutdown** command is issued on a T1 controller.

Syntax

```
shutdown
```

Syntax of the "no" Form

Use the *no* command to enable the interface after it is shut down:

```
no shutdown
```

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

sub-interface

This command starts configuration for a sub-interface on a serial interface. You can configure up to 50 sub-interfaces on the XSR.

Syntax

interface serial *interface_id.sub-interface_num* [**multi-point** | **point-to-point**

interface_id.sub-interface-num	The sub-interface, comprised of interface_num and numerical values. The entities are separated by a period "**.**" The number range is 1 to 50.
multi-point	The sub-interface acts as a multi-point connection, so that multiple DLCIs can be defined within this sub-interface to connect to multiple remote sites.
point-to-point	The sub-interface acts as a point-to-point connection.

Mode

Global configuration: **XSR(config)#**

Next Mode

Sub-interface configuration: **XSR(config-if<*xx*>)#**

Examples

This example selects sub-interface Serial *1/0.5* on a serial interface:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#encapsulation frame-relay
XSR(config-if<S1/0>)#no shutdown
XSR(config-if<S1/0>)#interface serial 1/0.5 multi-point
XSR(config-subif<S1/0.5>)#no shutdown
```

This example selects a sub-interface on a T1/E1 card:

```
XSR(config)#interface serial 2/1
XSR(config-if<S2/1>)#encapsulation frame-relay
XSR(config-if<S2/1>)#no shutdown
XSR(config-if<S2/1>)#interface serial 2/1:12.1 multi-point
XSR(config-subif<S2/1:12.1>)#no shutdown
```

Frame Relay Clear and Show Commands

clear frame-relay counter

This command clears the statistics of a particular Frame Relay DLCI, or all DLCIs under a specified Frame Relay sub-interface, or a Frame Relay port, or all Frame Relay ports on the XSR.

Syntax

clear frame-relay counter [[*interface*] [*interface*-num] [*dlci dlci-num*]]

interface -num	The interface or sub-interface number of the Frame Relay port or sub-interface affected by this command. If *interface serial interface-num* is not specified, then this command applies to all Frame Relay ports. If *interface-num* specifies a sub-interface, then only DLCIs in that particular sub-interface will be cleared. If *interface-num* calls for an interface, then all DLCIs on the Frame Relay interface will be cleared.
dlci-num	The specific DLCI whose statistics will be cleared.

Mode

EXEC: **XSR>**

clear frame-relay inarp

This command clears the inverse ARP table of one or all Frame Relay ports, causing the Frame Relay multipoint sub-interfaces to issue Inverse ARP requests to re-discover next hop addresses.

Syntax

clear frame-relay inarp [*interface*][*interface*-num][*dlci*] [dlci-num]

interface -num	If the *interface-num* or sub-interface number is set and the *dlci-num* is not, all learned inverse ARP entries for the interface and its logical sub-interfaces will be cleared.
dlci-num	The DLCI of a particular virtual port whose inverse ARP entry is to be cleared.

Mode

EXEC: **XSR>**

Examples

The following example clears all Frame Relay Inverse ARP entries:

```
XSR(config)#clear frame-relay inarp
```

This example clears all Frame Relay Inverse ARP entries for Interface *1/0* and its sub-interfaces:

```
XSR(config)#clear frame-relay inarp interface 1/0
```

The following example clears the Inverse ARP entry for DLCI *16* on sub-interface *1/0.1*:

```
XSR(config)#clear frame-relay inarp interface 1/0.1 dlci 16
```

show frame-relay fragment

This command displays information about Frame Relay fragmentation. When no parameters are specified, the output displays a summary of each data-link connection identifier (DLCI) configured for fragmentation including fragmentation type, configured fragment size, and number of fragments transmitted, received, and dropped. When a specific interface and DLCI are specified, additional details are displayed.

Syntax

show frame-relay fragment [**interface** *interface* [*dlci*]]

interface	A specific interface for which Frame Relay fragmentation data will be shown.
interface	Interface number containing the DLCI(s) for which to show fragmentation data.
dlci	Specific DLCI for which to display fragmentation data.

Mode

Privileged EXEC: **Router#**

Sample Output

The following is sample output from the command:

```
XSR(config)#show frame-relay fragment

Frame Relay End-to-End Fragmentation Summary
  interface          dlci       frag-size  in-frag    out-frag   dropped-frag
  Serial 2/0.1       960        53         0          0          0
  Serial 1/0:0.1      16        64         0          0          0
```

```
XSR(config)#show frame-relay fragment interface serial 2/0.1 960

Frame Relay End-to-End Fragmentation Detailed Statistics
 Serial 2/0.1 DLCI = 960    Fragment Size = 53
       Incoming Traffic                    Outgoing Traffic
  Fragmented pkts           = 0          Fragmented pkts          = 0
  Fragmented bytes          = 0          Fragmented bytes         = 0
  Assembled pkts            = 0          Pre-fragmented pkts      = 0
  Assembled bytes           = 0          Pre-fragmented bytes     = 0
  Non-fragmented pkts       = 0          Non-fragmented pkts      = 0
  Non-fragmented bytes      = 0          Non-fragmented bytes     = 0
  Dropped Assembled pkts    = 0          Interleaved pkts         = 0
  Pkt Sequence # Errors     = 0
  Unexpected Begin Frag     = 0
```

Parameter Descriptions

fragment-size	The configured fragment size in bytes.
In/out fragmented pkts	Sum of frames received/sent by this DLCI that had a fragmentation header.
In/out fragmented bytes	Sum of bytes, including those in the Frame Relay *bytes* headers, that have been received/sent by this DLCI.
In/out un-fragmented pkts	Sum of frames received/sent by this DLCI that do notrequire reassembly, and therefore do not contain the FRF.12 header. These counters can be incremented only when the end-to-end fragmentation type is set.
In/out un-fragmented bytes	Sum of bytes received/sent by this DLCI that do not require reassembly, and sp do not contain the FRF.12 header.
In assembled pkts	Sum of fully reassembled frames received by this DLCI, including frames without a Frame Relay fragmentationheader (in un-fragmented packets). This counter corresponds to frames viewed by upper-layer protocols.
In assembled bytes	Sum of bytes in the fully reassembled frames received by this DLCI, including frames without a Frame Relay fragmentation header (in un-fragmented bytes). This counter corresponds to the sum of bytes viewed byupper-layer protocols.
In dropped reassembled pkts	Sum of fragments received by this DLCI that are dropped for reasons such as running out of memory, receiving segments out of sequence, receiving an unexpected frame with a B bit set, or timing out on a reassembling frame.
Pkt Sequence # Error	Sum of fragments received by this DLCI that have an fragments unexpected sequence number.
Unexpected BeginFrag	Sum of fragments received by this DLCI that have an unexpected B bit set unexpected B (Begin) bit set. When this occurs, all fragments being reassembled are dropped and a new frame is begun with this fragment.
out pre-fragmented pkts	Sum of fully reassembled frames sent by this DLCI, including frames transmitted without a Frame Relay fragmentation header (out un-fragmented pkts).
out dropped fragmenting pkts	Sum of fragments dropped by this DLCI during transmission because of running out of memory.
in out-of-sequence fragments	Sum of fragments received by this DLCI with an unexpected sequence number.

in fragments with unexpected B bit set	Sum of fragments received by this DLCI that have an unexpected B (Begin) bit set. When this occurs, all fragments being reassembled are dropped and a new frame is begun with this fragment.
out interleaved packets	Sum of packets leaving this DLCI that have been interleaved between segments.

show frame-relay lmi

This command displays Local Management Interface (LMI) statistics. Enter the command without arguments to obtain statistics about all Frame Relay interfaces.

Syntax

show frame-relay lmi [*interface*] [*interface*-num]

interface -num	The interface or sub-interface number of the Frame Relayport or sub-interface affec ted by this command. If *interface serial interface-num* is not specified, then this command applies to all Frame Relay ports. If *interface-num* specifies a sub-interface, then only DLCIs in that particular sub-interface may be cleared. If *interface-num* calls for a port, then all DLCIs on the Frame Relay interface will be cleared.

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following example displays output on Serial interface *2/0* from an XSR with a Serial NIM installed:

```
XSR#show frame-relay lmi
LMI Statistics for Serial 2/0 (Frame Relay DTE)  LMI = AUTO (AUTO)
 Interface = INACTIVE
      Status Enq. Sent = 0                Status Msg. Rcvd = 0
        Status Timeout = 0              Updated Status Rcvd = 0
     # configured PVCs = 2              Invalid L2 LMI info = 0
local sequence number   = 127            net sequence number = 127
# PVCs reported by LMI = 0              Invalid L3 LMI Info = 0

 Down DLCIs:
  16, 18
```

The following example displays output on Serial interface *2/0:1* from an XSR with a T1/E1 Serial controller NIM installed:

```
LMI Statistics for Serial 0/2/0:1 (Frame Relay DTE)  LMI = NONE
 Interface = down
      Status Enq. Sent = 0                Status Msg. Rcvd = 0
        Status Timeout = 0              Updated Status Rcvd = 0
     # configured PVCs = 1              Invalid L2 LMI info = 0
local sequence number   = 127            net sequence number = 127
# PVCs reported by LMI = 0              Invalid L3 LMI Info = 0
```

Parameter Descriptions

LMI	The configured or auto-detected LMI type. If the port is set for AUTO LMI, then the XSR shows AUTO (nn), where nn is ILMI, ANSI, or ITU if the port has successfully negotiated/ detected the LMI supported by the switch, otherwise it displays AUTO.
Status Enq. Sent	Sum of LMI status enquiry messages sent.
Status Msgs Rcvd	Sum of LMI status messages received.
Status Timeouts	Sum of times the status message was not received within the keepalive time value.
Update Status Rcvd	Sum of LMI asynchronous update status messages received.
Invalid L2 LMI info	Sum of received LMI messages with invalid unnumbered information field.
Invalid L3 LMI	Sum of LMI messages with invalid fields.*fields*
Un-configured DLCIs	List of un-configured DLCIs are reported to be in an Active state by the Frame Relay switch. This field is not displayed if the configured LMI type is *None*.
Down DLCIs	List of configured DLCIs are reported to be in a Down or Inactive state by the Frame Relay switch. This field is not displayed if the configured LMI type is *None*.
Interface	*Down* marks the port as active but not communicating with the switch; *Inactive* marks the port as shut down; *Up* marks the port as operational.
Local/net sequence number	Value of current or next to transmit/received LMI control packet.

show frame-relay map

This command displays data from current frame-relay map entries.

Syntax

show frame-relay map

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following example displays a multi-point Frame Relay map:

```
XSR#show frame-relay map

Frame Relay Map Statistics (Serial 2/0)
  Serial 2/0.1  dlci  973 (0x3CD, 0xF0D0)  Remote Addr.  10.10.10.5,
      gratuitous-inverse-arp, bootp, static ip
  Serial 2/0.1  dlci  972 (0x3CC, 0xF0C0)  Remote Addr.  10.10.10.4,
      gratuitous-inverse-arp, static ip
  Serial 2/0.1  dlci  971 (0x3CB, 0xF0B0)  Remote Addr.  10.10.10.3,
      static ip
  Serial 2/0.1  dlci  970 (0x3CA, 0xF0A0)  Remote Addr.  un-resolved,
      gratuitous-inverse-arp
  Serial 2/0.1  dlci  960 (0x3C0, 0xF000)  Remote Addr.  un-resolved
```

The following example displays a point-to-point Frame Relay map:

```
XSR#show frame-relay map
Frame Relay Map Statistics (Serial 2/0)
  Serial 2/0.3  dlci  981 (0x3D5, 0xF450)  Remote Addr.  P2P,
      gratuitous-inverse-arp, bootp, static ip 2.2.2.3
```

Parameter Descriptions

Serial 2/0	Identifies a Frame Relay interface being displayed.
Serial 2/0.1	Identifies the specific sub-interface that is associated with a DLCI.
dlci 981(0x3D5,0xF450)	DLCI number displayed three ways: its *decimal* value, its *hexadecimal* value (0x3D5), and its value as it appears on the wire (0xF450).
Remote Addr.10.10.10.5	The remote peer IP address learned using Inverse ARP.
Remote Addr.	The node is waiting for Inverse ARP response to resolve *un-resolved* the remote peer's IP address.
Remote Addr.P2P	This DLCI does not require Inverse ARP to resolve the remote peer's IP address.
gratuitous Inverse-arp	This DLCI will offer a free Inverse ARP to help the remote learn changes to the local interface. The response from the remote is not used for address resolution.
bootp	This DLCI will respond to a broadcast bootp request originated from the adjacent peer. The bootp response includes the static IP address configured on this DLCI.
static ip 2.2.2.3	This DLCI has been configured with a static IP address for the remote peer. Inverse arp request will not be used to learn the remote's address.

show frame-relay pvc

This command displays statistics about permanent virtual circuits (PVCs) on Frame Relay interfaces. Statistics can be retrieved on specific Frame Relay interfaces by specifying the interface or the DLCI. Statistics on all PVCs can be shown by omitting arguments in the command.

If the LMI status report shows a PVC is not active, it is marked inactive. A PVC is marked deleted if it is not listed in a periodic LMI status message.

Syntax

show frame-relay pvc [**interface** *interface* [*dlci-num*]]

interface	Interface or sub-interface number containing the DLCI(s) for which you wish to display PVC information.
dlci	DLCI number used on the interface. Statistics for the specified PVC are displayed when a DLCI is also set.

Mode

Privileged EXEC: **XSR#**

Sample Output

```
XSR#show frame-relay pvc serial 2/0:1.1
```

```
PVC Statistics for Serial 2/0:1.1 (Frame Relay DTE)  LMI = NONE
       DLCI      = 16      PVC Status    = UP
INPUT:                      Pkt/Sec      = 0
   Packets       = 17941       Bytes     = 20018904       Drop Pkts     = 0
   BECN pkts     = 0        FECN pkts    = 0                 DE pkts    = 0
OUTPUT:                     Pkt/Sec      = 2
   Packets       = 17942       Bytes     = 20018904       Drop Pkts     = 0
   BECN pkts     = 0        FECN pkts    = 0                 DE pkts    = 0
 bcast pkts      = 0         bcast bytes = 0              CIR assists   = 0

PVC created: 12/01/2000 02:23:37   Last status change: 12/01/2000 02:23:47
   FRF.12 = ENABLED    Fragment size = 53
Adaptive Shape = DISABLED            Shaping Drops = 0
   minCIR=28000         BC=7000     BE=7000     limit=56   interval=125
```

Parameter Descriptions

DLCI	One of the Data-link Connection Identifier numbers for the PVC.
PVC STATUS	Status of the PVC: *ACTIVE* - DLCI is in data passing mode. *INACTIVE* - LMI message not received for longer than *n392dte* events and not in data passing mode. *DELETED* - LMI message declares DLCI is not activated.
Input: Pkt/Sec	The incoming data rate for this PVC in packets per second (measured for 8 seconds)
Input: pkts	Sum of packets received on this PVC.
Input: bytes	The packet rate in pps on this PVC in the last sampling period (last 8 seconds).
Input: Drop pkts	Sum of incoming packets on this PVC dropped.
In FECN pkts	Sum of packets received with FECN bit set.
In BECN pkts	Sum of packets received with BECN bit set.
In DE pkts	Sum of DE packets received.
Output: Pkt/Sec	Sum of packets sent on this PVC.
Output: pkts	Sum of packets sent on this PVC.
Output: bytes	Sum of bytes sent on this PVC.
Output: Drop pkts	Sum of outgoing packets on this PVC dropped.
Out BECN pkts	Sum of packets sent with BECN bit set. Value always 0.
Out FECN pkts	Sum of packets sent with FECN bit set. Value always 0.
Out DE pkts	Sum of DE packets sent. Value always 0.
Out bcast pkts	Sum of output broadcast packets. Value always 0.
Out bcast bytes	Sum of output broadcast bytes. Value always 0.
CIR assists	Sum of times the DLCI needed help to achieve CIR.
Pvc create time	Time the PVC was created.
Last status change	Time the PVC changed status (active to inactive).

FRF.12	FRF.12 has been disabled on this PVC. This line is not printed if disabled.
Fragment size	Size of the payload for fragmented packets.
Adaptive Shape	Status of Adaptive Shaping for this PVC.
Shaping Drops	Sum of packets dropped due to traffic shaping.
minCIR	The minimum Committed Information Rate, bits/sec.
BC	Current Committed burst size, in bits.
BE	Current Excess burst size, in bits.
Interval	Bc/CIR in milliseconds.

show frame-relay traffic

This command displays global Frame Relay statistics since the last reload.

Syntax

```
show frame-relay traffic
```

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

```
XSR#show frame-relay traffic
Frame Relay statistics:
TX: ARP requests = 19      ARP replies = 2
RX: ARP requests = 2       ARP replies = 19
```

show frame-relay map-class

This command displays Frame Relay map-class usage data. It provides a view of all configured Frame Relay map classes and whether they are being referenced by any Frame Relay interfaces.

Syntax

```
show frame-relay map-class
```

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#show frame-relay map-class

Total 7 frame relay map-classes configured in the node
"Central", "Branch_1", "three", "Class_4", "Class_5",
"Class_6", "test",

Map-Class "generic" has 1 registered users
```

```
 Serial 1/0, CIR= 64000, Bc=8000, BE= 9000, fragment=53
 Adaptive Shaping: Disabled, Service Policy: Voice

# FR Ports = 1, # FR sub-Interfaces = 3, # DLCIs = 7
```

show interface serial

The following statistics are added to the command if the port is configured for Frame Relay.

Sample Output

The following example displays T1 statistics:

```
XSR#show interface serial 2/0:1

********** Serial Interface Stats **********
Serial 2/0:1 is Admin Up
Internet address is not assigned

Frame Relay Port Statistics:
Line Protocol = UP
Encapsulation FRAME-RELAY IETF,  FRAME-RELAY DTE,  LMI = NONE
Num PVCs = 1, Total LMI Tx = 0, LMI Rx = 0
TX:  Packets = 18155, Bytes = 20214344 PPS = 0
RX:  Packets = 18154, Bytes = 20214072 PPS = 0
Approximate Speed = 128 Kbps    Discarded Packets TX/RX = 0/0
Sub Interface 1
      State = UP,  Num Stations = 1
      Configured DLCIs: 16, 18, 22
```

The following example displays Serial interface *2/0* statistics:

```
********** Serial Interface Stats **********
Serial 2/0 is Admin Up
Internet address is 10.10.11.30, subnet mask is 255.255.255.0

Frame Relay Port Statistics:
Line Protocol = UP
Encapsulation FRAME-RELAY IETF,  FRAME-RELAY DTE,  LMI = NONE
Num PVCs = 2, Total LMI Tx = 10, LMI Rx = 0
TX:  Packets = 10,  Bytes = 133   PPS = 0
RX:  Packets = 0,   Bytes = 0     PPS = 0
Approximate Speed = 65 Kbps    Discarded Packets TX/RX = 0/0
Sub Interface 1
 State = UP,  Num Stations = 1
 Configured DLCIs:
 16
Sub Interface 2
 State = UP,  Num Stations = 1
 Configured DLCIs:
 150

The name of this device is Ser2/0.
```

```
The card is 2.
The channel is 0.
The current MTU is 1506.
The device is in polling mode, and is active.
The last driver error is (null).
The physical-layer is HDLC-SYNC, the TX, RX clock source is external.
The device uses CRC-16 for Tx.
The device uses CRC-16 for Rx.
The type of encoding is NRZ.
The media-type is RS-232/V.28 (DTE).
The loopback mode is off.

Other Interface Statistics:
        ifindex              0
        ifType               23
        ifAdminStatus        1
        ifOperStatus         1
        ifLastChange         00:00:24
        ifInOctets           0
        ifInUcastPkts        0
        ifInNUcastPkts       0
         ifInDiscards         0
        ifInErrors           0
        ifInUnknownProtos    0
        ifOutOctets          173
        ifOutUcastPkts       10
        ifOutNUcastPkts      0
        ifOutDiscards        0
        ifOutErrors          0
        ifOutQLen            352
        RX overrun           0
```

10

Configuring the Dialer Interface

This chapter describes commands for the dialer, dialer backup, and Dial-on-Demand/Bandwidth-on-Demand services.

Observing Syntax and Conventions

The CLI command syntax and conventions use the notation described in the following table.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[x]	[ ] Square brackets indicate an optional parameter (*italic*)
[x | y | z]	[|] Square brackets with vertical bar indicate a choice of values
{x | y | z}	{ | } Braces with vertical bar indicate a choice of a required value
[x {y | z}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<*xx*>)	*xx* signifies interface type and number, e.g.: **F1, S2/1.0, D1, M57, L1, ATM0/1/1**
Next Mode entries display the CLI prompt after a command is entered	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

Dialer Interface Commands

The following set of commands defines dial services on the XSR:

- Dialer Interface Clear and Show commands on page 10-90.
- Dialer Backup commands on page 10-93.
- DOD/BOD commands on page 10-96.
- Dialer Watch commands on page 10-103.

dialer dtr

This command specifies that a non-V.25bis modem using Electronic Industries Association (EIA) signaling will be used on the serial line interface. This signal is known as the DTR signal. The **dialer string** command has no effect on DTR dialers. Be aware of the following mandatory conditions:

- The **dialer string** command must be set to the dialer interface that owns the dialer pool where the dialer DTR serial interface is added.
- The serial interface must be configured for synchronous data mode.
- The modem must be configured with DTR-controlled dialing interface, CTS follows DCD, DTR disconnects, sync data mode and a preset dialing out telephone number.

Syntax

dialer dtr

Syntax of the “no” Form

no dialer dtr

Default

DTR dialing is disabled

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

```
XSR(config-if<S1/1>)#dialer dtr
```

dialer pool

This command specifies which dialer pool the dialer interface should use. The dialer interface will use one of the physical interfaces in the dialer pool to attach to the interface's configured destination.

Syntax

dialer pool *number*

number	Dial pool number, ranging from 1 to 255.

Syntax of the “no” Form

no dialer pool

Default

Disabled - no pool is specified.

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Note: This command is intended for dialer interfaces only.

Example

The following example shows dialer interface 0 assigned to dialer pool 6.

```
XSR(config)#interface dialer 0
XSR(config-if<D1>)#dialer pool 6
XSR(config-if<D1>)#no shutdown
```

dialer pool-member

This command configures physical interfaces for dial devices only.

Syntax

dialer pool-member *number* [**priority** *priority*]

number	Dialpool number ranging from 1 to 255.
priority	Priority of the interface within the dialing pool - ranging from 0 (lowest) to 255 (highest). Ports with the highest priority are selected first for dialing out.

Syntax of the “no” Form

no dialer pool-member *number*

Defaults

- Disabled. When enabled, no default dialing pool number is assigned
- Priority: 0
- Minimum: 0
- Maximum: 255

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example shows a serial interface belonging to two dialer pools with priorities configured for each pool:

```
XSR(config-if)#interface serial 1/0
XSR(config-if<S1/0>)#dialer pool-member 1 priority 10
XSR(config-if<S1/0>)#dialer pool-member 2 priority 20
XSR(config-if<S1/0>)#no shutdown
```

dialer string

This command creates a string used to place a call a destination or subnet. Typically, it is the telephone number needed for dialing.

Syntax

dialer string *dial-string* [**class** *class-name*]

dial-string	Phone number to be sent to a dial device.
class-name	Map class associated with this dialer string.

Syntax of the "no" Form

no dialer string *dial-string*

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

This example shows that dialer interface 0 configured to use map-class *XXX* when using dialer string *9055559988*:

```
XSR(config-if)#interface dialer 0
XSR(config-if<D0>)#dialer string 9055559988 class XXX
```

dialer wait-for-carrier-time (interface configuration)

This command configures the time a dialer interface waits for a carrier signal. It is used when configuring a particular dialer interface.

Syntax

dialer wait-for-carrier-time *seconds*

seconds	Interval the interface waits for a carrier signal when a call is placed via the dial device.

Syntax of the "no" Form

The *no* form of this command resets to default value:

no dialer wait-for-carrier-time

Default

60 seconds

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example specifies a wait time of 90 seconds for the carrier signal on serial port 1/0:

```
XSR(config-if<S1/0>)#dialer wait-for-carrier-time 90
```

dialer wait-for-carrier-time (map-class dialer configuration)

This command configures the time to wait for a carrier signal associated with a specific dialer map class. Dialer map classes are used to configure certain characteristics with dialer strings when configuring dialer ports.

Syntax

dialer wait-for-carrier-time *seconds*

seconds	Interval the port waits for a carrier signal when a call is placed through the dial device.

Syntax of the "no" Form

The *no* form of this command resets to the default value:

```
no dialer wait-for-carrier-time
```

Default

60 seconds

Mode

Map-class dialer configuration: **XSR(config-map-class)#**

Example

The example below specifies a 120-second wait time for the carrier signal of the dialer map class *TEST* on Dialer port 57:

```
XSR(config-if<D57>)#interface dialer 57
XSR(config-if<D57>)#ip address 196.16.25.1 255.255.255.0
XSR(config-if<D57>)#encapsulation ppp
XSR(config-if<D57>)#dialer remote-name SiteA
XSR(config-if<D57>)#dialer string 4165555584 class TEST
XSR(config-if<D57>)#dialer pool 1
XSR(config)#map-class dialer TEST
XSR(config-map-class)#dialer wait-for-carrier-time 120
```

interface dialer

This command adds a dialer interface to connect with one or more specified sub-networks. A dialer interface connects to a dial device via a pool of physical ports.

The dialer interface is created in two ways: point-to-point or point-to-multipoint by using the multipoint parameter. When configured, the dialer line is not physically connected but the entry is maintained in the routing table thus preserving on-demand access when interesting packets are received and accepted by an Access Control List (ACL).

This mode of operation of the dialer interface is called spoofing and it is the default mode for this interface. Spoofing mode changes to non-spoofing mode when the following conditions are met:

- Another interface or sub-interface is set with the **backup interface dialer** command.
- The interface configured with the **backup** command (the primary interface) is up.

Dial-on-demand applications require that a *dialer-group*, *dialer-list* and *ACL* also be configured.

Syntax

interface dialer [*number* | *multi-point*][*sub-interface*]

number	Non-spoofed mode for a backup line or spoofed mode for on-demand connectivity to a remote peer. Dialer interface number ranges from 0 to 255.
multi-point	Spoofed, point-to-multi-point mode configuring on-demand connectivity to remote peers.
sub-interface	Sub-interface of the dialer interface.

Syntax of the "no" Form

The *no* form of this command removes the dialer interface:

interface dialer *number*

Mode

Global configuration: **XSR(config)#**

Next Mode

Dialer Interface configuration: **XSR(config-if<D>)#**

Default

Interface is spoofed

Examples

The following example configures Dialer port *200* in backup mode with minimal settings:

```
XSR(config)#interface dialer 200
XSR(config-if<D200>)#ip address 200.17.10.5 255.255.255.0
XSR(config-if<D200>)#encapsulation ppp
XSR(config-if<D200>)#authentication chap
XSR(config-if<D200>)#no shutdown
```

The following example configures the dialer in point-to-point spoofed mode with interesting packets defined by ACL *101*, a dialer-group and associated dialer-list mapped to ACL *101*:

```
XSR(config#access-list 101 permit ip 0.0.0.0 255.255.255.255 255.255.255.255 0.0.0.0
XSR(config)#interface dialer 3
XSR(config-if<D3>)#dialer-group 7
XSR(config)#dialer-list 7 protocol ip list 101
```

The following example configures the dialer in multi-point spoofed mode with interesting packets defined by ACL *101*, a dialer-group and associated dialer-list mapped to ACL *101*:

```
XSR(config)#interface dialer 3 multi-point
```

```
XSR(config-if<D3>)#dialer-group 7
XSR(config-if<D3>)#access-list 101 permit ip 0.0.0.0 255.255.255.255 255.255.255.255
0.0.0.0
XSR(config)#dialer-list 7 protocol ip list 101
```

map-class dialer

This command defines the dial string's characteristics and associates them with a unique class name. Once the **map-class dialer classname** command is executed the parameters assigned to that *classname* must be configured. The *classname* assigned must match the *classname* assigned to the dialer *string* class *classname* so they can be linked.

Syntax

map-class dialer *classname*

classname	Unique class identifier.

Default

None - no class name

Mode

Global configuration: **XSR(config)#**

Next Mode

Map-Class Dialer configuration: **XSR(config-map-class<*xx*>)#**

Example

The example below specifies a *90*-second wait time for the carrier signal of the dialer map class *TEST* on Dialer port *0*:

```
XSR(config)#interface dialer 0
XSR(config-if<D0>)#ip address 196.16.25.1 255.255.255.0
XSR(config-if<D0>)#encapsulation ppp
XSR(config-if<D0>)#dialer remote-name sitea
XSR(config-if<D0>)#dialer string 4165555584 class TEST
XSR(config-if<D0>)#dialer pool 1
XSR(config-if<D0>)#no shutdown
XSR(config)#map-class dialer 57
XSR(config-map-class<57>)#dialer wait-for-carrier-time 90
```

modem-init-string

This command sets an AT command string used to initialize a modem.

Syntax

modem-init-string *word*

word	Text to initialize the modem.

Syntax of the "no" Form

The *no* form of this command removes the modem-init-string:

```
no modem-init-string
```

Mode

Map-Class Dialer configuration: **XSR(config-map-class<*xx*>)#**

Example

The following example specifies a modem initialization string to disable dialtone detection for the Map Class *Remote*:

```
XSR(config-map-class<Remote>)#modem-init-string ATX3
```

Dialer Interface Clear and Show Commands

clear dialer

This command clears dialer statistics for physical interfaces connected to the dialer interfaces. If the interface is not specified, all interface (for the dialer) statistics will be cleared.

Syntax

```
clear dialer
```

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#clear dialer
```

show dialer

This command displays general information and some configurations of interfaces configured under the dialer; for instance, the dialer interfaces and the serial and async interfaces under the dialer interfaces.

Syntax

show dialer [*number*]

number	Interface number.

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#show dialer 1
```

Sample Output

The following is sample output from the **show dialer** command for a dialer interface:

```
#show dialer 5

Dialer5
Dialer state is: UP
Wait for carrier default: 60, default retry: 3
Dial String     Success Failures        Map Class
3200            2       0

Dialer pool 23
(Serial 2/0:0, )
```

Parameter Descriptions

Dialer1	Name of the dialer interface.
Wait for carrier(30 secs)	Seconds to wait for carrier signal.
Default retry	Number of default call retries.
Dial String	Dial strings to used to make calls.
Successes	Number of successful connections.
Failures	Failed Connections.
Map Class	Name of associated map class.
Dialer pool 2, priority 0	Indicates that this interface is a member of dialer pool 2 with a priority of 0 in that pool.
Serial0	Type of interface.

show dialer maps

This command displays dialer policies.

Syntax

```
show dialer maps
```

Mode

EXEC: **XSR#**

Sample Output

The following is sample output from the **show dialer maps** command:

```
Dialer maps configured on Interface <Dialer1>:
 Next hop IP address: <10.10.10.2>
 Remote host: <robo2>
 Map class: <isdn>
```

```
 Phone numbers: <2400:12>
 Connection speed/type: <64k>/<On Demand>

Dialer maps configured on Interface <Dialer2>:
 Next hop IP address: <20.20.20.2>
 Phone numbers: <2400>
 Connection speed/type: <not set>/<On Demand>
```

show dialer sessions

This command displays information regarding dialer sessions.

Syntax

show dialer sessions

Mode

EXEC: **XSR#**

Sample Output

The following is sample output from the **show dialer sessions** command:

```
XSR#show dialer sessions
ID     Interface     Type          State         MLPPP  Phone#    Phys Intf
0001   Dialer1       On Demand     IDLE          001
0002   Dialer1       Multilink     CONNECTED     001    3100      Serial 2/0:30
0003   Dialer1       Incoming      CONNECTED     001              Serial 2/0:12
0004   Dialer0       On Demand     WAITING       000    2600      D-Serial 1/0:0
```

Parameter Descriptions

ID	Dial session ID number - node-wide and unique. Range: 1 to 512.
Interface	Dialer interface number which has requested the dial session.
Type	Dial session type: • *On demand*: session that handles on demand connection requests • *Backup:* session which is requested by a backed up interface or watched route • *Multilink:* dial session requested by a multilink group used for backup or on demand • *Callback:* dial callback session. • *Bandwidth:* a Bandwidth on Demand requested connection.
State	IDLE, WAITING, CALLING or CONNECTED.
MLPPP	MLPPP group number to which the dial session belongs.
Phone No	Number used to dial out.
Phys Intf	Dialer pool port used to build a switched link with remote peer.

Dial Backup Commands

The following set of commands defines a backup dial line.

backup

This command set backup functionality on Serial, Ethernet or sub-interfaces. You can also specify a delay before a secondary interface is brought up or down after a primary interface is brought up or down. We suggest this command be used when lines suffer intermittent disruptions causing the primary line to come up and fall temporarily. A backup delay ensures the secondary line does not come up and down prematurely.

Note: The XSR sets **UTC** for *time-range* calculation.

Syntax

backup *interface dialer dialer-interface-number* [**delay** *enable-delay disable-delay* [**never**]][**time-range** *hh:mm hh:mm*]

interface	Dialer interface number used for backup.
delay *enable-delay disable-delay*	Backup *enable* delay, ranging from 0 to 99999999, followed by the backup *disable* delay, ranging frm 0 to 99999999, or the keyword **never** indicating the backup, once enabled, is not being disabled when the primary link comes up. The *enable-delay* is the interval in seconds that elapses after the primary port goes down. The *disable-delay* is the interval in seconds that elapses after the primary port comes up.
never	Stops the secondary port from being deactivated.
time-range *hh:mm hh:mm*	Backup timer range - start from hh:mm to end hh:mm. When backup is not set, it is is always active. Otherwise it is active during the configured time range only.

Syntax of the “no” Form

The *no* form of this command removes backup from the interface:

no backup *interface*

Default

1 second

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example provides a 10-second delay in activating the secondary line and a 20-second delay in deactivating the secondary line when the primary serial line goes up and down.

```
XSR(config)#interface serial 1/1
XSR(config-if<S1/1>)#backup delay 10 20
XSR(config-if<S1/1>)#no shutdown
```

backup interface dialer

This command designates a Serial or Fast/GigabitEthernet/GigabitEthernet interface or sub-interface as a backup dialer interface.

Caution: To configure a backup FastEthernet/GigabitEthernet interface or sub-interface, the port *must* be in the shutdown state.

Syntax

backup interface dialer *number*

number	Dialer interface number to use as the backup interface. Range: 0 to 255.

Syntax of the "no" Form

no backup interface dialer *number*

Note: Only one dialer interface can be associated with one dialer pool but one dialer pool may be associated with many dialer interfaces.

Default

Disabled

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Examples

The example below configures Dialer interface 57 as the backup for Fast/GigabitEthernet port 2:

```
XSR(config)#interface fastethernet 2
XSR(config-if<F2>)#backup interface dialer 57
XSR(config-if<F2>)#ip address 192.168.27.114 255.255.255.0
XSR(config-if<F2>)#no shutdown

XSR(config)#interface serial 1/2
XSR(config-if<S1/2>)#physical-layer async
XSR(config-if<S1/2>)#dialer pool-member 1
XSR(config-if<S1/2>)#no shutdown

XSR(config)#interface dialer 57
XSR(config-if<D57>)#dialer pool 1
XSR(config-if<D57>)#dialer redial attempts 3 forever
XSR(config-if<D57>)#dialer string 67921
XSR(config-if<D57>)#encapsulation ppp
XSR(config-if<D57>)#ip address 10.10.10.1 255.255.255.0
XSR(config-if<D57>)#no shutdown
```

Ethernet backup is applied further in the example below where Dialer interface 57 is configured as the DSL backup (PPPoE) for Fast/GigabitEthernet sub-interface 2.1 - invoking the sub-interface enables PPPoE. Note that the IP address of the PPPoE caller is negotiated over PPP and the MTU size is reset to 1492 bytes to avoid Web access problems by PCs attached to the XSR.

```
XSR(config)#interface fastethernet 2
XSR(config-if<F2>)#no shutdown

XSR(config)#interface fastethernet 2.1
XSR(config-if>)#backup interface dialer 57
XSR(config-if>)#encapsulation ppp
XSR(config-if>)#ip address negotiated
XSR(config-if>)#ip mtu 1492
XSR(config-if>)#no shutdown
```

backup time-range

This command configures a period when the backup dialer should be up and down, regardless of traffic on the line. A backup dialer port is configured to protect a primary interface and once its time-range is specified, the backup dialer port can be enabled and disabled.

Syntax

backup time-range *start-time end-time*

start-time	Time in *hh:mm* when the dialer port should be enabled.
end-time	Time in *hh:mm* when the dialer port should be disabled.

Syntax of the “no” Form

The *no* form of this command disables the time-range feature:

no backup time-range

Default

None

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Examples

The example below configures Dialer port 1 to be enabled at *6:30 a.m.* and to disable itself at *11:55 p.m.*

```
XSR(config)#interface serial 1/1
XSR(config-if<S1/1>)#backup interface dialer 1
XSR(config-if<S1/1>)#no shutdown
XSR(config-if<S1/1>)#backup time-range 06:30 23:55
```

show interface dialer

This command displays general information for a dialer interface.

Syntax

show interface dialer *number*

number	Dialer interface number ranging from 0 to 255

Mode

Privileged EXEC: **XSR#**

Sample Output

The example below displays output from the **show interface dialer** command:

```
XSR#show interface dialer

********** Dialer Interface Stats **********
Dialer1 is Admin Up
Internet address is 10.10.10.1, subnet mask is 255.255.255.0

Dialer1
Dialer state is: UP
Wait for carrier default: 60, default retry: 3
Dial String            Success Failures        Map Class

Dialer pool 3
(Serial 2/0:0, )
Free pool ISDN channels: <25>
Free pool serial ports: <0>

Neighbor Dial String       Success       Failures        Map Class
3100                       1             0

Active links MLPPP group <1> to <10.10.10.2>: <5>
```

DOD/BOD Commands

The XSR supports the following Dial on Demand (DoD)/Bandwidth on Demand (BoD) commands.

dialer-group

This command controls dialer access by configuring an interface to belong to a specific dialing group. This access group is associated with an access list by the **dialer-list** command.

Packets which match the dialer group trigger a connection request. That is, the destination address of packets is evaluated against one or more ACLs; if the packets pass, either a call is initiated (if no connection were already established) or the idle timer is reset (if a call is active).

Syntax

dialer-group *group-number*

group-number	Number of the dialer access group to which the specified interface belongs. Acceptable values are nonzero, positive integers between 1 and 10.

Syntax of the 'no' Form

Use the *no* form of this command to remove an interface from the specified dialer access group:

no dialer-group

Mode

Dialer Interface configuration: **XSR(config-if<D*x*>)#**

Example

The following example configures dialer group 7 on dialer interface 1, mapping ACL 101 to dialer-list 7:

```
XSR(config)#interface dialer 1
XSR(config-if<D1>)#dialer-group 7
XSR(config)#access-list 101 permit ip 0.0.0.0 255.255.255.255 255.255.255.255 0.0.0.0
XSR(config)#dialer-list 7 protocol ip list 101
```

dialer-list

This command defines a dialer list to control dialing by protocol or by a combination of protocol and Access Control List (ACL). Because IP is the sole protocol supported at this time, an ACL *must* be specified using the **dial-list** command.

Syntax

dialer-list *dialer-group* **protocol** *protocol-name* **list** *access-list-number*]

dialer-group	Number of a dialer access group identified in any **dialer-group** command, ranging from 1 to 10.
protocol-name	Only the protocol **ip** is supported at this time.
list	Specifies that an access list will be used for defining a granularity finer than an entire protocol.
access-list-number	Numbers specified in IP standard (1 - 99) or extended (100 - 99) access lists.

Syntax of the "no" Form

Use the *no* form of this command to delete a dialer list:

no dialer-list *dialer-group* [**protocol** *protocol-nam* [**list** *access-list-number*]

Mode

Global configuration: **XSR(config)#**

Example

The following example maps ACL 1350 to dialer list 57:

```
XSR(config)#access-list 57 permit ip 0.0.0.0 255.255.255.255 255.255.255.255 0.0.0.0
XSR(config)#dialer-list 57 protocol ip list 1350
```

dialer called

This command maps incoming calls to one of the dialer interfaces. A maximum number of 32 called numbers per dialer interface can be configured.

Syntax

dialer called *DNIS:subaddress*

DNIS:subaddress	Dialed Number Identification Service, or the called party number, a colon, and the ISDN subaddress.

Syntax of the "no" Form

The *no* form of this command removes the configured number:

no dialer called *DNIS:subaddress*

Mode

Dialer Interface configuration: **XSR(config-if<Dx>)#**

Example

The following example configures a dialer profile for a receiver with DNIS *12345* and ISDN subaddress *6789*:

```
XSR(config)#interface dialer 1
XSR(config-if<D1>)#dialer called 12345:6789
```

dialer caller

This command configures caller ID screening with an option providing ISDN callback. The XSR will accept calls from a specified phone number. A maximum of 32 caller numbers can be set per dialer port.

The command matches numbers starting with the least significant digits of the calling number, starting from the last digit. Typically the ISDN switch does not provide the complete calling number, only the local number (four to seven of the least significant digits).

The dialed number must be configured in the Dialer interface.

Syntax

dialer caller *number* [*callback*]

number	Phone number to screen. Limit: 32 characters.
callback	Returns the call to the dialer. This option applies to DoD applications and supports PPP and MLPPP. If used in a backup capacity, set the number of retries to 1.

Note: If the ISDN switch does not provide the calling number, callback will fail.

Syntax of the "no" Form

The *no* form of this command disables the feature:

no dialer caller *number*

Mode

Dialer Interface configuration: **XSR(config-if<D*x*>)#**

Example

The following example configures the dialer caller numbers to screen:

```
XSR(config)#interface dialer 1
XSR(config-if<D1>)#dialer caller 5084712345
```

dialer idle-timeout

This command specifies the idle timeout interval before the XSR disconnects the line. The timeout trigger is based on outbound traffic only.

Caution: This command *must* be invoked on the *called* side of a link with a *0* value to ensure the link is not dropped after 120 seconds by the called side.

Syntax

dialer idle-timeout *seconds*

seconds	Interval before disconnecting the line, ranging from 0 to 2,147,483 seconds. Specifying 0 disables the timeout.

Syntax of the "no" Form

Use the *no* form of this command to reset the idle timeout to the default:

no dialer idle-timeout

Mode

Dialer Interface configuration: **XSR(config-if<D*x*>)#**

Default

120 seconds

Examples

The following example resets the idle-timeout:

```
XSR(config)#interface dialer 1
XSR(config-if<D1>)#dialer idle-timeout 300
```

The following example disables the idle-timeout:

```
XSR(config-if<D1>)#dialer idle-timeout 0
```

dialer map

This command configures a Dialer or Integrated Services Digital Network (ISDN) interface to call one or multiple sites. Each dialer interface can be configured with a maximum of 16 different dialer maps. The command also enables spoofing on the specified dialer interface but is available in *multi-point* mode only.

Syntax

All options are shown in the first form of the command as follows:

dialer map *protocol next-hop-address* [**name** *hostname*][**class** *map-class*][**spc**] [**speed 56** | **64**][*dial-string*][*:isdn-subaddress*]]

protocol	Protocol keyword; **ip** is supported at this time.
next-hop-address	Protocol address used to match against addresses to which packets are destined.
name	The remote system with which the local router or access server communicates.
hostname	Case-sensitive name or ID of the remote device (usually the host name).) It is used for incoming call mapping based on the authenticated user name negotiated under PPP.
map-class	Name of map class used to dial out.
spc	A Semi-Permanent Connection between your equipment and the exchange.
speed 56 | 64	Keyword and value indicating the line speed in kilobits per second to use. For ISDN only.
dial-string	Telephone number sent to the dialing device when it recognizes packets with the specified next hop address that matches the access lists defined.
:isdn-subaddress	Sub-address number used for ISDN multipoint connections.

Syntax of the “no” Form

The *no* form of this command deletes a particular dialer map entry:

no dialer map *protocol next-hop-address* [**name** *hostname*] [**class** *map-class*] [**spc**] [**speed 56** | **64**] [**broadcast**] [*dial-string*[*:isdn-subaddress*]]

Mode

Dialer Interface configuration: **XSR(config-if<D*x*>)#**

Default

Speed: 64 kbps

Example

The following example configures a next hop IP address, SPC, hostname and line speed for map class *AcmeMap*:

```
XSR(config)#dialer map 1
XSR(config-if<Dx>)#dialer map ip 192.168.57.9 class AcmeMap name AcmeHost spc
speed 56 12345:6789
```

dialer persistent

This command brings up a permanent switched connection in the absence of an interesting packet or primary-line-down backup dial trigger.

Syntax

dialer persistent [**delay** *n*]

n	Interval that the dial-out process is delayed after the Dialer interface boots up, ranging from 1 to 2147483 seconds.

Syntax of the "no" Form

The *no* form of this command deletes the persistent setting:

no dialer persistent

Mode

Dialer Interface configuration: **XSR(config-if<D*x*>)#**

Default

-1 second

Example

The following example configures Dialer interface 57 to be persistent for two minutes:

```
XSR(config)#interface dialer 57
XSR(config-if<D57>)#dialer persistent 120
```

dialer redialer attempts

This command sets the redial trigger after failed dial attempts. With an infinite number of specified redial attempts, it is possible to physically connect a modem at any time after setting the dial trigger and still make a connection. Also, if more resources (interfaces) are available in the dialer pool, the dialer is free to redial all members of the pool.

Syntax

dialer redial attempts *n* **interval** *m* **re-enable** *t* [*forever*]

attempts	Redial attempts.

n	Number of redial attempts made if dial-up or ISDN connection establishment fails, ranging from 1 to 65535.
interval	Period between redial attempts.
m	Interval period, ranging from 5 to 2678400 seconds (31 days).
re-enable	Period for which the port is disabled if all redial tries fail.
t	Re-enable period, ranging from 5 to 2678400 seconds.
forever	Number of redial attempts applied to all members of the dialer pool in a never-ending loop if dial-up or ISDN connection establishment is unsuccessful. Redial attempts end if the dial trigger is reset or the connection is established.

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Defaults

- Attempt: 1 (no redial)
- Interval: 10 seconds
- Re-enable: 5 seconds

Example

Assuming you have configured Serial interfaces 1/0, 1/1, and 1/2 as part of dialer pool 1, the following example sets the dialer to attempt dialing each interface five times (if all attempts are unsuccessful), indefinitely until the dial trigger is reset or a connection is successfully established.

```
XSR(config)#interface dialer 1
XSR(config-if<D1>)#dialer pool 1
XSR(config-if<D1>)#dialer redial attempts 5 forever
```

dialer remote-name

This command specifies, for a dialer interface, the PPP authenticated user name of the remote router that is calling in.

Syntax

dialer remote-name *username*

username	Case-sensitive character string identifying the remote device with a maximum length of 255 characters.

Mode

Dialer Interface configuration: **XSR(config-if<D*x*>)#**

Example

The following example sets the authentication name for the remote router on Dialer interface *7:*

```
XSR(config)#interface dialer 7
XSR(config-if<D1>)#dialer remote-name "Auth West"
```

Dialer Watch Commands

dialer watch-group

This command enables Dialer Watch backup on a dialer interface with up to 16 watch-groups.

Note: The XSR sets **UTC** for *time-range* calculation.

Syntax

dialer watch-group *group-number*

group-number	Assigned number that will point to a globally defined list of IP addresses to watch, ranging from 1 to 255.f

Syntax of the "no" Form

Use the *no* form of this command to disable this feature:

no dialer watch-group *group-number*

Mode

Dialer Interface configuration: **XSR(config-if<D*x*>)#**

Example

The following example configures a dialer watch group:

```
XSR(config-if<D3>)#dialer watch-group 57
```

dialer watch-list

This command adds a list of IP addresses you want monitored. Use this command with the dialer watch-group interface configuration command. The number of the group list must match the group number.

Note: The XSR sets **UTC** for *time-range* calculation.

Syntax

dialer watch-list *group-number* [**delay route-check initial** *initial-delay*][**delay connect** *connect-delay*][**delay disconnect** *disconnect-delay*][**ip** *ip-address address-mask*]][**time-range** *start-time end-time*]

group-number	Number assigned to the list, ranging from 1 to 255.
ip	IP is the only routed protocol supported for Dialer Watch at this time.
ip-address	IP address or address range to be applied to the list.

address-mask	IP address mask to be applied to the list.
initial-delay	The delay interval between the time when a new route is added to any dialer watch list and the start of the backup process for that route if the route fails to come up. This delay prevents the XSR from starting backup process for the configured watched routes immediately after bootup. Range: 1 to 2,147,483 seconds.
connect-delay	The delay interval between when a route set up under the watch list goes down and when the dialer subsystem starts the backup process. Range: 1 to 2,147,483 seconds.
disconnect-delay	The delay interval between when a route set up under the watch list and currently backed up comes up and when the dialer subsystem ends the backup process. Range: 1 to 2,147,483 seconds.
start-time *end-time*	Time range when the watch-list is set as active using the 24-hour format *hh:mm* for both start and the end times. The watch-list does not trigger the backup outside this time range regardless of the state of route collection.

Syntax of the "no" Form

Use the *no* form of this command to disable this feature:

no dialer watch-list *group-number* [**delay route-check initial** *initial-delay*][**delay connect** *connect-delay*][**delay disconnect** *disconnect-delay*][**ip** *ip-address address-mask*]

Mode

Dialer Interface configuration: **XSR(config-if<D*x*>)#**

Default

- Initial delay: 30 seconds
- Connect delay: 2 seconds
- Disconnect delay: 2 seconds

Example

The following example configures the dialer watch option:

```
XSR(config-if<D9>)dialer watch-list 57 delay route-check initial 15 delay connect
1 delay disconnect 1 ip 192.168.69.9 255.255.255.0
```

Sample Output

The following is sample output from the **show interface dialer** command displaying dialer watch statistics:

```
********** Dialer1 Interface Stats **********
Internet address is 1.1.1.2, subnet mask is 255.255.255.0

Dialer1 is Admin Up, Description: <Vancouver>
Oper Status is SPOOFING
Dial stats:
 wait for carrier 60s, redial attempts 3, redial interval 10s
```

```
 dial string: 3200, success: 0, fail: 0
Dialer pool 1 stats:
 member: Serial 1/3:0,
 available B-channels: 30,  serial ports: 0
Watch-group stats:
 watch-group 1, rt cnt 1, trigg cnt 1, state is UP,
  delays: init 10, connect 3, disconnect 3,
  time range 10:15  11:15
  timer expires in 18h:32m:28s
 watch-group 2, rt cnt 1, trigg cnt 1, state is UP,
  delays: init 30, connect 60, disconnect 2,
  time range 10:0  11:17
  timer expires in 18h:17m:29s
```

11

ISDN BRI and PRI Commands

Observing Syntax and Conventions

The CLI Syntax and conventions use the notation described in the following table.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[*x*]	[ ] Square brackets indicate an optional parameter (*italic*)
[*x* | *y* | *z*]	[|] Square brackets with vertical bar indicate a choice of values
{*x* | *y* | *z*}	{ | } Braces with vertical bar indicate a choice of a required value
[*x* {*y* | *z*}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<***xx***>)	*xx* signifies the interface type and number; e.g., **`F1`**, **`G3`**, **`S2/1.0`**, **`M57, BRI-1/1`**, **`PRI-2/1. F`** indicates a FastEthernet, and **`G`** a GigabitEthernet interface.
Next Mode entries display the CLI prompt after a command is entered	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

ISDN Commands

The following set of commands allows you to configure BRI/PRI functionality on the XSR.

interface bri

This command configures a BRI interface for each physical BRI port on the BRI NIM card. When entered, the **`interface bri`** command must be followed by the **`isdn switch-type`** command for BRI ISDN applications, *or* the **`leased-line bri`** command for BRI leased line applications. If none of the above commands are issued BRI ports are non-operational.

Syntax

`interface bri` *board/slot/port*

`board/slot/port`	BRI board, slot, and port numbers. For leased-line applications: 1 for B1 and 2 for B2. Sub-ports are added by the **`leased-line [56 | 64]`** command.

Syntax of the "no" Form

no interface bri *board/slot*

Mode

Global configuration: **XSR(config)#**

Next Mode

BRI Interface configuration: **XSR(config-if<BRI-*xx*>)#**

Example

The following example acquires BRI B-channel 1 interface mode:

```
XSR(config)#interface bri 1/1
XSR(config-if<BRI-1/1>)#
```

isdn answer1, isdn answer2 (BRI)

This command, **isdn answer1**, directs the XSR to screen a called-party or sub-address number in the incoming setup message for ISDN BRI calls. Issue the **isdn answer1** or **2** command to filter incoming calls based on the called-party or sub-address number.

If you do not specify this command, all calls are processed or accepted. If you specify the command, the XSR must verify the incoming called-party number and the sub-address before processing and/or accepting the call. The verification proceeds from right to left for the called-party number; it also proceeds from right to left for the sub-address number.

You can configure the called-party number only or the sub-address. In such a case, only the configured value is verified. To configure a sub-address only, include the colon (:) before the sub-address number.

Note: This command is applicable to the BRI ETSI switch only.

Syntax

isdn answer1 [*called-party-number*][*:subaddress*]

called-party-number	Telephone number of the called party. At least one value, *called-party-number* or *subaddress*, must be specified. This value can total no more than 50 digits. Number that follows as a sub-address. The colon (:) sets both *called-party* and *subaddress*, or *subaddress* only. *called-party* and *subaddress*, or *subaddress* only.
subaddress	Sub-address number used for ISDN multipoint connections. At least one value, *called-party* or *subaddress*, must be set. The sub-address can total no more than 50 digits.

Syntax of the "no" Form

Use the *no* form of this command to remove the verification request:

no isdn answer1 [*called-party-number*][*:subaddress*]

Default

No verification of either number

Mode

BRI Interface configuration: **XSR(config-if<BRI-*xx*>)#**

Examples

The following example configures BRI interface 1/1 with called-party and sub-address numbers:

```
XSR(config)#interface bri 1/1
XSR(config-if<BRsaI-1/1>)#isdn answer1 6171234:5678
```

The following example configures BRI interface 2/0 with a *sub-address* number only:

```
XSR(config)#interface bri 2/0
XSR(config-if<BRI-2/0>)#isdn answer1:5678
```

isdn bchan-number-order (PRI)

This command configures an ISDN PRI interface to choose an outgoing call in either ascending or descending order. The XSR selects the lowest or highest available B-channel starting at either channel B1 (ascending) or channel B23 for a T1 anD-channel B30 for an E1 (descending). Use this command only if your service provider requests it to decrease the probability of call collisions.

Syntax

isdn bchan-number-order {*ascending* | *descending*}

ascending	Selects the outgoing B-channel in ascending order as follows: 1 to 24 for a T1 and 1 to 31 for an E1 card.
descending	Selects the outgoing B-channel in descending order as `follows`: 24 to 1 for a T1 and 31 to 1 for an E1 card.

Syntax of the "no" Form

To restore the default, use the *no* form or simply reconfigure the interface with the new value:

no isdn bchan-number-order

Default

Descending

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example sets the T1 controller to make call selections in ascending order:

```
XSR(config)#controller t1 1/0/0
XSR(config-controller<T1-1/0:0>)#description "T1 at Acme"
XSR(config-controller<T1-1/0:0>)#framing esf
XSR(config-controller<T1-1/0:0>)#linecode b8zs
XSR(config-controller<T1-1/0:0>)#pri-group
XSR(config-controller<T1-2/1>isdn bchan-number-order ascending
```

isdn call

This command is used for debugging purposes *only* to test call setup procedures with a Central Office ISDN switch or test equipment. It is automatically disconnected after 30 seconds.

Note: Enter this command in Privileged EXEC, not Global configuration mode.

Syntax

isdn call c/p [*board/slot/port*] *dialing-string* [**56** | **64**]

c/p	BRI or PRI port ID.
dialing-string	Called phone number and sub-address.
56	Call placed at 56 kbps rate.
64	Call placed at 64 kbps rate.

Mode

Privileged EXEC: **XSR#**

Example

The following example initiates an ISDN call on BRI port 2/1 at 56 kbps:

```
XSR#isdn call 2/1:61712345678 56
<186>Jul 28 22:49:51 10.10.10.20 ISDN:
No Channel Available For Test Call
```

isdn calling-number

This command configures an ISDN PRI or BRI interface to include caller-number in the out-going setup message. This billing number is used for non Fully Initializing Terminals (FIT) outside North America only because the **isdn spid1/2** command already configures the LDN.

A PRI or BRI port can have only one ISDN calling-number entry. For ISDN PRI, this command is intended for use when the network offers better pricing on calls in which devices present the caller number. When configured, the calling number is included in the outgoing setup message.

Note: There is no mechanism to mark outgoing calls with the Calling Number and Calling Subaddress for call routing on the *receiving* end.

Syntax

isdn calling-number *calling-number:subaddress*

calling-number	Number of the device making the outgoing call. Only one entry is allowed.
:subaddress	Extension of the phone number.

Syntax of the "no" Form

no isdn calling-number

Mode

BRI Interface configuration: **XSR(config-if<BRI-x>)#**

Example

The following example specifies a calling number for the XSR:

```
XSR(config)#interface bri 1/0
XSR(config-if<BRI-1/0>)#isdn calling-number 5088781234
```

isdn disconnect

This command is used for debugging purposes to test ISDN connectivity. It sets up an ISDN data call to test call setup procedures with a Central Office ISDN switch or test equipment. It is used to disconnect a call before automatic disconnect occurs in 30 seconds or if a call is not dropped.

Note: Enter this command in Privileged EXEC, not Global configuration mode.

Syntax

isdn disconnect c/p *channel_number*

c/p	BRI or PRI port ID.
channel_number	BRI: 1 or 2, E1 PRI: 0 to 31, T1 PRI: 0 to 22.

Mode

Privileged EXEC: **XSR#**

Example

The following example sets up a test call on channel 24 on BRI port 1/1:

```
XSR#isdn disconnect 1/1 24
<186>Jul 28 22:49:51 10.10.10.20 ISDN:
No Channel Available For Test Call
```

isdn spid1, isdn spid2 (BRI)

This command specifies the Service Profile Identification Number (SPID) which is supplied by your ISDN service provider.

North America (NOAM) ISDN switches use Fully Initializing Terminals (FIT) service which require the CPE to register its SPID with the Central Office (CO) before service can begin.

Syntax

isdn spid1 *spid-number* {**max digits**| *ldn*} {**max digits**}
isdn spid2 *spid-number* {**max digits**} *ldn*} {**max digits**}

spid-number	Number of the service to which you have subscribed, up to 26 digits. Assigned by the ISDN service provide, it is a 7 to 10-digit phone number with additional prefix and suffix digits such as 905361707001. If a SPID is set to 0 and the no isdn autodetect command was issued (autodetect not active), then the line is assumed to be No FIT type and will not attempt registration with the CO.
ldn	This Local Directory Number is a 7 or 10-digit number assigned by the service provider. It is also used for setting the calling number for outgoing calls.

Syntax of the “no” Form

The *no* form of this command removes the SPID number:

no isdn spid1 {max digits| *ldn***} {max digits}**
no isdn spid2

Mode

BRI Interface configuration: **XSR(config-if<BRI-*x*>)**

Example

The following example specifies a SPID and LDN for the B1 channel:

```
XSR(config-if<BRI-2/1>)#isdn spid1 508876123401 5088761234
```

isdn switch-type (BRI/PRI)

This command sets the central office switch type for the ISDN port, and triggers the creation of the following three dedicated serial interfaces: *slot/card/port:0*, *slot/card/port:1* and *slot/card/port:2* for the D, B1 and B2 channels, respectively. Because this command does not have a no form, you can only replace the switch with another, not remove it. The **show interface** command displays the ISDN interface status.

Note: This command is valid only *after* the **pri-group** command was issued.

Syntax

isdn switch-type *switch-type* {*basic-dms100* | *basic-ni1* | *basic-ntt* | *basic-net3* | *primary-net5* | *primary-ni2* | *primary-5ess* | *primary-dms100* | *primary-ntt*}

BRI Switch Types:

basic-dms100	North America legacy ISDN switch.
basic-ni1	National ISDN 1 switch for North America.
basic-5ess	North America legacy ISDN switch: *not supported*.
basic-ntt	Switch for ISDN in Japan.
basic-net3	ETSI-compliant switch for Euro-ISDN.

PRI Switch Types:

primary-net5	ETSI-compliant switch for Euro-ISDN.
primary-ni2	T1 National ISDN switch type (T1 default).
primary-5ess	T1 NOAM legacy switch.
primary-dms100	T1 NOAM legacy switch.
primary-ntt	T1/J1 ISDN switch for Japan.

Syntax of the “no” Form

The *no* form of this command deletes the three serial interfaces:

no isdn switch-type

Defaults

- BRI: *basic-net3*
- PRI: *primary-net5*
- E1: *primary-net5*
- T1: *primary-ni2*
- J1: *primary-ntt*

Mode

BRI/PRI Interface configuration: **XSR(config-if<BRI/PRI-xx>)#**

Example

The following example selects a switch type on the BRI 1/1 interface:

```
XSR(config)#interface bri 1/1
XSR(config-if<BRI-1/1>)#isdn switch-type basic-net3
```

leased-line bri

This command sets up an ISDN BRI port for leased-line operation. Leased-line service at 64 or 128 kbps via BRI is provided in Japan and Germany. The 56 and 112 kbps speeds are provided for eventual North American deployment of this service.

Once a BRI interface is configured for access over leased lines, it is no longer a dialer interface, and signaling over the D-channel no longer applies. Although the interface is called interface BRI, it is configured as a synchronous serial port and all serial port commands are available.

This command creates a serial interface that is configured as a standard serial port. It can be issued once for speeds equal to and higher then 112 as both B-channels are bound to the created serial interface. For 56 and 64 bps speeds, the command can be issued twice to create individual serial interfaces :1 and :2 for B1 and B2, respectively.

After you enter the command, you must exit BRI configuration mode and configure the channels by entering **`interface bri [board/port:1]`** or **`interface bri [board/port:2]`**. These Bearer ports are configured as regular synchronous serial interfaces.

Note: The **`shutdown/no shutdown`** *`channel`* commands are overridden by the *interface bri* **`shutdown/no shutdown`** commands.

Syntax

`leased-line bri` *`speed`* `{56 | 64 | 112 | 128 | 144}`

`56 | 64`	Two streams are supported, one on each B-channel.
`112 | 128 |144`	One stream is supported over the bonded B1 + B2 or B1+ B2 + D-channels.

Syntax of the “no” Form

The *no* form of this command cancels leased-line BRI by deleting the earlier created serial interface and returning to the *basic-net3* ISDN switch-type:

```
no leased-line bri
```

Default

CMD/switch type *basic-net3*

Mode

BRI Interface configuration: **`XSR(config-if<xx>)#`**

Examples

The following example configures two data streams on leased-line BRI interface 1/1 at 56 kbps with PPP encapsulation:

```
XSR(config)#interface bri 1/1
XSR(config-if<BRI-1/1>)#leased-line 56
XSR(config)#interface bri 1/1:1
XSR(config-if<BRI-1/1:1>)#ip address 1.1.1.2 255.255.255.0
XSR(config-if<BRI-1/1:1>)#encapsulation ppp
```

The following example configures BRI B-channel 2:

```
XSR(config)#interface bri 1/1:2
XSR(config-if<BRI-1/1:2>)#ip address 1.1.1.3 255.255.255.0
XSR(config-if<BRI-1/1:2>)#encapsulation frame-relay
```

The following example configures one data stream on leased-line BRI interface 1/1 at 112 kbps with Frame Relay encapsulation:

```
XSR(config)#interface bri 1/1
XSR(config-if<BRI-1/1>)#leased-line 112
XSR(config)#interface bri 0/1/2:1
XSR(config-if<BRI-1/2:1>)#ip address 1.1.1.3 255.255.255.0
XSR(config-if<BRI-1/2:1>)#encapsulation frame-relay
```

pri-group

This command configures a T1/E1 port to ISDN PRI on a channelized E1/T1 card. All 23 T1 or 30 E1 time slots are assigned to ISDN control.

Syntax

```
pri-group
```

Syntax of the “no” Form

The *no* form of this command deregisters the T1/E1 controller from the ISDN controller. Use the *no* form to remove the ISDN PRI and restore the T1/E1 controller to its default mode:

```
no pri-group
```

Mode

Controller configuration: **`XSR(config-controller<T/Exx>)#`**

Example

The following NFAS example configures PRI with D-channel backup:

```
XSR(config)#controller t1 1/0
XSR(config-controller<T1-1/0>)#pri-group
```

shutdown (BRI)

This command forces all data calls to be disconnected and signals all internal XSR resources that the port is not available.

Syntax

shutdown [*board/slot/port*]

board/slot/port	XSR board, slot and port numbers.

Syntax of the "no" Form

no shutdown [*board/slot/port*]

Mode

Interface configuration: **XSR(config)#shutdown**

ISDN Debug and Show Commands

debug isdn

This command initiates a Layer 2 or 3 ISDN debug session to trace failed calls at the D channel level. Issuing the command has the effect of locking out debugging by any other Telnet or Console connection. If both Layer 2 (Q921) and 3 (Q931) choices are selected, tracing will display *both* layers.

Note: To prevent unauthorized personnel from observing the debug session on the network, users with *privilege level 15* only can issue this command.

You can exit the debug session either by isuing the **no debug isdn** command or terminating the Telnet or Console session.

Optionally, you can set a limit of up to 9999 messages which will display at the CLI after which the debug session will end. If the limit is *not* specified, after 100 displayed messages, the **no debug isdn** command will automatically be run. The **limit** parameter is a global value that is refreshed each time **debug isdn** is entered.

Syntax

debug isdn *slot/card/port* **Q931** | **Q921** [**limit** {*10-9999*}]

slot/card/port	ISDN board, slot, and port numbers.
Q931	Layer 3 protocol tracing enabled for a port issue.
Q921	Layer 2 protocol tracing enabled for a port issue.
limit	ISDN debug session exits after all messages display.
10-9999	Number of messages displayed during a debug session.

Syntax of the "no" Form

The *no* form of this command removes ISDN message tracing. You may choose to issue the command with all or no parameters selected.

no debug isdn *slot/card/port* **Q931** | **Q921** [**limit** {*10-9999*}]

Default

Messages: 100

Mode

EXEC Configuration: **XSR**

Examples

The following example configures Layer 3 ISDN debugging on the specified interface:

```
XSR#debug isdn 0/1/0 q931
ISDN-DBG 0/1/0 Enable Q931 Tracing
```

show controllers bri

This command displays physical line data concerning Basic Rate Interface (BRI) sub-interfaces.

Syntax

show controllers bri [*board/slot/port*] [*:channel number*]

board	XSR board, slot and port numbers: *<1-2>/<0-1>*
/slot	Card, port and D-channel or, *<1-2>/<0-1>:<0-2>*
/port	Card, port and channel (0 = D, 1 = B1, 2 = B2).

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Example

The following output is produced for BRI sub-interface 2/1:0

```
XSR#show controllers bri 2/1:0
Forward Engine Serial Layer Tx/Rx Stats:

RX FROM UPPER LAYER & TX TO DRIVER
        Pcks Rx         = 0
        Pcks Tx         = 0
        Pcks Discarded = 0
RX FROM DRIVER & TX TO UPPER LAYER
        Pcks Rx         = 0
        Pcks Tx         = 0
        Pcks Discarded = 0
```

```
Packet Processor Tx Scheduler Stats:
        0 Packet driver Tx OK
        0 Packet driver not Tx: MUX END_ERR_BLOCK
        0 Packet driver not Tx: MUX ERROR
        0 Packet driver not Tx: Unknown Msg from MUX

The unit number is 167772177.
The interrupt number is 27.
General: SCC 4 parm ram = 0xa0290f00, reg = 0xa0291660

TX RING ENTRIES:
        The data ring starts at 0xa0290200.
        TxDRNum = 16, pTxMblkDR = 0x010fc120, TxDRIdx = 0
        TxDRCleanIdx = 0

(-2)    CmdStsLen 0x00000000, pBuf 0x00000000
(-1)    CmdStsLen 0x00000000, pBuf 0x00000000
( 0)    CmdStsLen 0x00000000, pBuf 0x00000000
( 1)    CmdStsLen 0x00000000, pBuf 0x00000000
( 2)    CmdStsLen 0x00000000, pBuf 0x00000000
        [...]

RX RING ENTRIES:
        The data ring starts at 0xa02901c0.
        RxDRNum = 8, pRxMblkDR = 0x00ffd200, RxDRIdx = 0
        RxBuffSize = 1728, RxBuffOffset = 160

(-2)    CmdStsLen 0x80000000, pBuf 0x21e146e0
(-1)    CmdStsLen 0xa0000000, pBuf 0x21e14da0
( 0)    CmdStsLen 0x80000000, pBuf 0x21e11e60
( 1)    CmdStsLen 0x80000000, pBuf 0x21e12520
( 2)    CmdStsLen 0x80000000, pBuf 0x21e12be0
        [...]
```

show interface bri

This command displays the status of the B and D-channels' serial driver. Generally speaking, BRI channels are displayed exactly as standard serial ports and PRI channels are displayed as standard T1/E1/ISDN-PRI serial channels.

If the B-channel is not connected by an active call, the OPER state will be down. The D-channel will display L1 and L2 status in addition to standard output.

To display the D- or B-channels use the following commands:

- BRI - **`show interface bri 1/0`** or **`show interface bri 1/0:0`** - for D channel
- PRI - **`show interface serial 2/1:23`** for T1 D channel
- PRI - **`show interface serial 2/1:15`** for E1 D channel
- PRI - **`show interface serial 2/1:0 - 22`** for T1 B channels

- PRI - **show interface serial 2/1:0 - 14, 16-30** for E1 B channels

Use the following table for reference.

Table 11-1 Channel Number Mappings

	Service Provider Channel Numbering		Enterasys Channel Numbering	
	B-channels	D-channels	B-channels	D-channel
T1	1-23	24	0-22	23
E1	1-31	16	0-30 (not 15)	15
BRI	1, 2	-	1, 2	0

Syntax (PRI)

show interface bri [*card/port*]:[*channel number*]

:channel	Valid channel numbers are: *E1* - 0 to 30 (D-channel: 15),
number	*T1* - 0 to 22 (D-channel: 23)

Syntax (BRI)

show interface bri [*card/port*]:[*channel number*]

channel number	1 and 2 (0 is the D-channel)

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following output is displayed for the BRI interface 1/1:0:

```
********** Serial Interface Stats **********
D-Serial 1/1:0 is Admin Up / Oper Down

********************** ISDN  Stats ISDN-BRI  1/1 *********************
Layer 1: DOWN  Layer 2: DOWN  State: OFFLINE      Admin Up        Oper Down

Term. 1 Spid:2200555 State:  OFFLINE Cause: 000
Term. 2 Spid:2201555 State:  OFFLINE Cause: 000
Total Length = 257
The name of this device is bri1/1/0.

The card is 1.
The port is 1.
The following output is displayed for the PRI interface 2/1:
********************** ISDN  Stats ISDN-PRI  21 *********************
Layer 1: UP Layer 2: UP State: ONLINE      Admin Up        Oper Up
```

Standard output of the command follows but is not displayed here.

The following output is displayed for the BRI interface 2/1:

```
XSR#sh interface bri 2/1

********** Serial Interface Stats **********
D-Serial  2/1:0 is Admin Down / Oper Down

********************** ISDN  Stats ISDN-BRI  2/1 ******************
Layer 1: DOWN  Layer 2: DOWN  State: OFFLINE      Admin Down      Oper Down

The name of this device is bri2/1/0.

The card is 2.
The port is 1.
The channel is 0.
The current MTU is 1506.
The device is in polling mode, and is INACTIVE.
The channel is logically INACTIVE.
The operational state is OPER_DOWN.
The protocol used is LAPD.
The baud rate is 16000 bits/sec.
The device uses CRC-16 for Tx.
The device uses CRC-16 for Rx.
Other Interface Statistics:
        ifindex              0
        ifType               75
        ifAdminStatus        1
        ifOperStatus         2
        ifLastChange         00:00:00
        ifInOctets           0
        ifInUcastPkts        0
        ifInNUcastPkts       0
        ifInDiscards         0
        ifInErrors           0
        ifInUnknownProtos    0
        ifOutOctets          0
        ifOutUcastPkts       0
        ifOutNUcastPkts      0
        ifOutDiscards        0
        ifOutErrors          0
        ifOutQLen            16
```

show isdn history

This command displays past ISDN actions on the XSR.

Syntax

show isdn history [*board/slot/port*]

board/slot/port	XSR board, slot and port numbers.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following output displays incoming and outgoing call data for BRI interface 1/0 and its sub-interfaces:

```
XSR#show isdn history 1/0
********************** ISDN Call History ISDN-BRI  1/0 **********************
    Channel        Dir       Start Time     End Time   Cause   Phone Num
BRI  1/0:2       INCOMING   07:23:10:135  07:23:40:158 016               2100
BRI  1/0:1       OUTGOING   07:23:09:817  07:23:39:983 016               2100
BRI  1/0:1       OUTGOING   06:32:21:351  06:32:24:947 016               2100
BRI  1/0:1       OUTGOING   06:31:09:214  06:31:11:804 016               2100
BRI  1/0:1       INCOMING   06:31:00:856  06:31:02:296 016    No CALLING Num.
BRI  1/0:1       INCOMING   06:24:59:093  06:25:03:116 016    No CALLING Num.
BRI  1/0:2       INCOMING   06:21:03:982  06:21:07:906 016               2100
BRI  1/0:1       OUTGOING   06:21:03:719  06:21:07:906 016               2100
```

The following output displays incoming call data for PRI interface 2/0 and sub-interfaces 23 - 30:

```
XSR#show isdn history 2/0

********************** ISDN Call History ISDN-PRI  2/0 **********************
    Channel        Dir       Start Time     End Time   Cause   Phone Num
Serial 2/0:30    INCOMING   20:15:33:888  20:15:51:276 016    No CALLING Num.
Serial 2/0:29    INCOMING   20:15:33:874  20:15:51:142 016    No CALLING Num.
Serial 2/0:28    INCOMING   20:15:33:880  20:15:51:047 016    No CALLING Num.
Serial 2/0:27    INCOMING   20:15:33:870  20:15:50:924 016    No CALLING Num.
Serial 2/0:26    INCOMING   20:15:33:866  20:15:50:835 016    No CALLING Num.
Serial 2/0:25    INCOMING   20:15:33:860  20:15:50:709 016    No CALLING Num.
Serial 2/0:24    INCOMING   20:15:33:856  20:15:50:621 016    No CALLING Num.
Serial 2/0:23    INCOMING   20:15:33:853  20:15:50:486 016    No CALLING Num.
```

Prameter Descriptions

Cause	Cause code describing why the call was disconnected.
Phone Num	Calling number for incoming calls and called number for outgoing calls.

show isdn active

This command displays current call information of all BRI or PRI ports, or only the selected port specified by board/slot/port identifier.

Syntax

show isdn active [*board/slot/port*]

board/slot/port	XSR board, slot and port numbers.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following output displays current call data on BRI interface 1/0:

```
XSR#show isdn active 1/0

************************* ISDN  Stats ISDN-BRI  1/0 **********************
Layer 1:   UP  Layer 2:   UP  State: ONLINE       Admin Up        Oper Up

Ch No   State     Dir   Speed    Called / Start    Calling / Destination  Cause
 1   CONNECTED OUTGOING  64 2100                                          0
                            07:27:52:314          Outgoing Test Call
 2   CONNECTED INCOMING  64 2100                  2100                    0
                            07:27:52:686          Unknown Call
```

Parameter Descriptions

Call Type	Type of call: INCOMING for incoming, OUTGOING for outgoing or **--** when call direction cannot be determined.
Calling or Called Phone	Number for outgoing call displays.
10 least significant digits of called number	8 least significant digits of called sub-address.
The following parameters are for incoming call displays:	
10 least significant digits of calling number	8 least significant digits of the calling sub-address. If the incoming SETUP message does carries the relevant information element, nothing will be printed.
Destination	Specifies the Dialer interface/Dialer session that handles the call. The name display is limited to 10 characters.
Speed	56 or 64.
B/S/P	Port ID Board/Slot/Port.
Cause	3-digit number from 0 to 127 sent by the CO in a Cause Information Element. Refer to the table in the *Configuring ISDN* chapter of the *XSR User's Guide* for Cause Code explanations.
Start	Call start date and time.
End	Call end time.

show isdn service

This command displays the service status of all or selected ISDN ports.

Syntax

show isdn service [*board/slot/port*]

board/slot/port	XSR board, slot and port numbers.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Parameter Descriptions

Layer 1 Status	ACTIVE| DEACTIVE | PENDING - (Active - cable up and line synchronized)
Layer 2 Status	LAPD: UP | DOWN; State: OFFLINE (Offline - ISDN is not registered with SPIDs or SPIDs not required)

Examples

The following example displays statistics fom the BRI NOAM port:

```
XSR#show isdn service 1/1

********************** ISDN  Service ISDN-BRI  1/1 ********************
Layer 1:   UP  Layer 2:   UP  State: ONLINE       Admin Up        Oper Up

Term. 1 Spid:2200555 State:   ONLINE Cause: 000
Term. 2 Spid:2201555 State:   ONLINE Cause: 000
Ch No  State   Ch No  State   Ch No  State   Ch No  State   Ch No  State
 1    IDLE      2    IDLE
```

The following example shows output from BRI port 1/0:

```
#show isdn service 1/0 (BRI)

********************** ISDN  Service ISDN-BRI  1/0 ********************
Layer 1:   UP  Layer 2:   UP  State: ONLINE       Admin Up        Oper Up

Ch No  State    Ch No  State    Ch No  State    Ch No  State    Ch No  State
 1    IDLE      2    IDLE
```

The following example shows output from PRI port 2/0:

```
XSR#show isdn service 2/0

********************** ISDN  Service ISDN-PRI  2/0 ********************
Layer 1:   UP  Layer 2:   UP  State: ONLINE       Admin Up        Oper Up

Ch No  State   Ch No  State   Ch No  State   Ch No  State   Ch No  State
 0 CONNECTED    1 CONNECTED    2 CONNECTED    3 CONNECTED    4 CONNECTED
 5 CONNECTED    6 CONNECTED    7 CONNECTED    8 CONNECTED    9 CONNECTED
10 CONNECTED   11 CONNECTED   12 CONNECTED   13 CONNECTED   14 CONNECTED
15 D-channel   16 CONNECTED   17 CONNECTED   18 CONNECTED   19 CONNECTED
```

```
20 CONNECTED    21 CONNECTED    22 CONNECTED    23 CONNECTED    24 CONNECTED
25 CONNECTED    26 CONNECTED    27 CONNECTED    28 CONNECTED    29 CONNECTED
30 CONNECTED
```

12

Configuring Quality of Service

Observing Syntax and Conventions

The CLI Syntax and conventions use the notation described in the following table.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[x]	[ ] Square brackets indicate an optional parameter (*italic*)
[x | y | z]	[|] Square brackets with vertical bar indicate a choice of values
{x | y | z}	{ | } Braces with vertical bar indicate a choice of a required value
[x {y | z}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<*xx*>)	*xx* signifies the interface, class map, policy map or other value you specify; e.g., **F1**, **G3**, **S2/1.0**, **<Your Name>**. **F** indicates a FastEthernet, and **G** a GigabitEthernet interface.
Next Mode entries display the CLI prompt after a command is entered.	
Sub-command headings are displayed in ***red*** text.	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

QoS Commands

The following set of commands configure Quality of Service (QoS) values for the XSR:

- "Policy-Map Commands" on page 12-84.
- "Class-map Commands" on page 12-101.
- "QoS Show Commands" on page 12-105.

service-policy

This command attaches a policy map to an output or input interface. You can attach a single policy map to one or more interfaces.

Syntax

service-policy [**input** | **output**] *policy-map-name*

policy-map-name	Attaches the specified policy map onto the output port.

Syntax of the "no" Form

The *no* form of the command removes a policy map from the interface:

no service-policy [*input* | *output*]

Mode

Interface configuration: **XSR(config-if<xx>)#**

Example

The following example associates policy map *ACMEpolicy* with Serial *1/0*:

```
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#service-policy output ACMEpolicy
```

Policy-Map Commands

policy-map

This command creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy. Sub-commands associated with this command are:

- **bandwidth** - Specifies the bandwidth allocated for a class belonging to a policy map. Go to page 12-86 for the command definition.
- **class** - Specifies the criteria for classifying traffic. Go to page 12-87 for the command definition.
- **police** - Configures traffic policing. Go to page 12-89 for the command definition.
- **priority** - Prioritizes a class of traffic belonging to a policy map. Go to page 12-90 for the command definition.
- **queue-limit** - Specifies the peak number of packets the queue can hold for a class policy configured in a policy map. Refer to page 12-91 for the command definition.
- **random-detect** (RED) - Enables Random Early Detect on an interface. Refer to page 12-92 for the command definition.
- **random-detect** (WRED) - Enables Weighted Random Early Detect on an interface. Refer to page 12-93 for the command definition.
- **random-detect dscp** - Specifies the DSCP value. Refer to page 12-93 for the command definition.

- **random-detect exponential-weighting-constant** - Configures the WRED exponential weight factor for the average queue size calculation.Refer to page 12-95 for the command definition.
- **random-detect precedence** - Configures WRED minimum and maximum threshold and maximum drop probability values for a IP precedence value. Go to page 12-96 for the command definition.
- **set cos** - Marks the IEEE 802.1 priority in the header of output VLAN packets with a Class of Service (CoS) matching clause. Go to page 12-97 for the command definition.
- **set ip dscp** - Marks a packet by setting the IP Differentiated Services Code Point (DSCP) parameter. Go to page 12-98 for the command definition.
- **set ip precedence** - Sets the precedence value in the IP header. Go to page 12-99 for the command definition.
- **shape** - Enables and configures traffic shaping on a class. Go to page 12-100 for the command definition.

Use the **policy-map** command to specify the name of the policy map to be created, added to, or modified before you can configure policies for classes whose match criteria are defined in a class map. Invoking the **policy-map** command enables QoS Policy-Map configuration mode in which you can configure or modify the class policies for that policy map.

You can configure class policies in a policy map only if the classes have match criteria defined for them. You use the **class-map** and **match** commands to configure the match criteria for a class. You can configure up to 64 class policies in a policy map.

A single policy map can be attached to multiple interfaces concurrently. If you attempt to attach a policy map to an interface and available bandwidth on the interface cannot accommodate the total bandwidth requested by class policies comprising the policy map, the interface becomes oversubscribed. In such a case, when classes try to send with all of their bandwidth, some classes may be unable to transmit.

Whenever you modify class policy in an attached policy map, CBWFQ is notified and the new classes are installed as part of the policy map in the CBWFQ system.

Syntax

policy-map *policy-map-name*

policy-map-name	Name of the policy map.

Syntax of the “no” Form

Use the *no* form of this command to delete a policy map:

no policy-map *policy-map-name*

Mode

Global configuration: **XSR(config)#**

Next Mode

Policy-Map configuration: **XSR(config-pmap-<*xx*>)#**

Example

These commands create class-map *class1* and define its match criteria:

```
XSR(config)#class-map class1
XSR(config-cmap<class1>)#match access-group 136
```

These commands create the policy map which is defined to contain policy specifications for *class1* and the *default* class:

```
XSR(config)#policy-map policy1
XSR(config-pmap<policy1>)#class class1
XSR(config-pmap-c<class1>)#bandwidth 2000
XSR(config-pmap-c<class1>)#queue-limit 40
XSR(config-pmap<policy1>)#class class-default
XSR(config-pmap-c<class-default>)#queue-limit 20
```

bandwidth

This command specifies or modify the bandwidth allocated for a class belonging to a policy map. It is used in conjunction with a class defined by the **class-map** command. The **bandwidth** command specifies the bandwidth for traffic in that class. Class-Based Weighted Fair Queueing (CBWFQ) derives the weight for packets belonging to the class from the bandwidth allocated to the class. CBWFQ then uses the weight to ensure that the queue for the class is serviced fairly.

The amount of bandwidth can be specified in percentages or kilobits per second (kbps). When configured in *kbps*, the class weight is calculated as a ratio of the bandwidth specified for that class over the available link bandwidth. The available link bandwidth is equal to the interface bandwidth minus the sum of all bandwidth reserved for low latency queues. When configured in *percentages*, the class weight is equal to the bandwidth percentages.

Configuring bandwidth in percentages is most useful when the underlying link bandwidth is unknown, changes over time, or the relative class bandwidth distributions are known. For interfaces that have adaptive shaping rates, CBWFQ can be set by configuring class bandwidths in percentages.

The following restrictions apply to the **bandwidth** command:

- If the **percent** keyword is used, the sum of class bandwidth percentages cannot exceed 100% .
- The amount of bandwidth set should be large enough to also accommodate Layer 2 overhead.
- A policy map can have all the class bandwidths specified in kbps or all the class bandwidths specified in percentages, but not a mix of both. But, the unit for the **priority** command in the priority class can be different from the bandwidth unit of the CBWFQ.

Note: When the bandwidth of an interface is insufficient to satisfy the bandwidth of a policy map, the interface becomes oversubscribed and some CBFWQ classes may become unable to transmit.

Syntax

bandwidth {*bandwidth-kbps* | **percent** *percent*}

bandwidth-kbps	Amount of bandwidth, in kbps, assigned to the class.
percent	Available bandwidth percentage assigned to the class.

Syntax of the "no" Form

Remove the bandwidth specified for a class by using the *no* form of this command:

```
no bandwidth
```

Mode

Policy-Map Class configuration: **XSR(config-pmap-c<*xx*>)#**

Example

The following example specifies a bandwidth of *2000* Kbps for *polmap6*:

```
XSR(config)#policy-map polmap6
XSR(config-pmap<polmap6>)#class acl22
XSR(config-pmap-c<acl22>)#bandwidth 2000
XSR(config-pmap-c<acl22>)#queue-limit 30
```

class

This QoS policy-map sub-command specifies the name of the traffic class whose policy you want to create or to change and sets the criteria for classifying traffic. The XSR provides a robust set of matching rules for you to define the criteria.

Before using the **class** command, you must first enter the **policy-map** command to identify the policy map you want to change. This also allows you to enter QoS policy-map configuration mode. After you specify a policy map, you can configure policy for new classes or modify policy for any existing classes in that policy map.

The class name you specify in the policy map ties the characteristics for that class - that is, its policy - to the class map and its match criteria, as configured using the **class-map** command.

When a class is removed, available bandwidth for the interface is incremented by the amount previously allocated to the class.

Note: The XSR supports a maximum of 64 traffic classes.

The predefined default class called *class-default* is the class to which traffic is directed if that traffic does not satisfy the match criteria of other classes whose policy is defined in the policy map.

Syntax

class {*class-name* | *class-default*}

class-name	Specifies the name of the class to set or modify policy.
class-default	Specifies the default class to configure or modify policy.

Note: *Class-default* cannot be removed with the **no class** command.

Syntax of the "no" Form

The *no* form of this command removes a class from the policy map:

no class {*class-name*}

Mode

Policy-Map configuration: **XSR(config-pmap<*xx*>)#**

Next Mode

Policy-Map Class configuration: **XSR(config-pmap-c<*xx*>)#**

Example

This example creates *class1* with a minimum of *20* percent in the event of congestion, and the queue reserved for this class can enqueue *40* packets before tail drop is enacted to handle additional packets.

```
XSR(config)#policy-map policy1
XSR(config-pmap-policy1>)#class class1
XSR(config-pmap-c<class1>)#bandwidth percent 20
XSR(config-pmap-c<class1>)#queue-limit 40
```

These commands create *class2* with a minimum of *3000* kbps of bandwidth for this class in the event of congestion. RED drops up to one out of three packets when the average queue size becomes bigger than *34* and drops each packet if it becomes bigger than *57*. RED packet drop is used for congestion avoidance.

```
XSR(config-pmap<policy1>)#class class2
XSR(config-pmap-c<class2>)#bandwidth 3000
XSR(config-pmap-c<class2>)#random-detect 34 57 3
```

These commands configure the *default* map class where a maximum of *20* packets per queue are enqueued before tail drop is enforced to handle additional packets.

```
XSR(config-pmap<policy1>)#class class-default
XSR(config-pmap-c<class-default>)#queue-limit 20
```

clear policy-map

This command removes Policy Map statistics for specified interfaces.

Syntax

clear policy-map interface *type number*

type	XSR interface type: BRI, Dialer, Fast/GigabitEthernet, Loopback, Multilink, and Serial.
number	Card, port, channel, and sub-interface number.

Mode

EXEC: **XSR>** or **XSR(config)#**

police

This command configures traffic policing.

Syntax

police bps [*burst-normal*][*burst-max*][**conform-action** *action*][**exceed-action** *action*][**violate-action** *action*]

bps	Average rate ranging from 1,000 to 100,000,000 bps.
burst-normal	Normal burst size ranging from 1,000 to 51,200,000 bps. If less than 1000 bytes *burst-normal* will be set to 1000 bytes.
burst-max	Excess burst size ranging from 1,000 to 51,2000,000 bytes. Value **must** be greater than or equal to *normal-burst* size. It will automatically be changed to the *normal-burst* size if less than *normal-burst*.
conform-action	Action to take on packets that conform to the rate limit.
exceed-action	Action to take on packets that exceed the rate limit.
violate-action	Action to take on packets that violate normal and maximum burst sizes. If *violate-action* is set, the token bucket algorithm will use two token buckets.
action	Action to take on packets. You may specify *one* keyword: • **drop** - Drops the packet. • **set-prec-transmit** *new-prec* - Sets IP precedence and sends the packet. • **set-dscp-transmit** *new-prec* - Sets the differentiated services code point (DSCP) value and sends the packet. • **transmit** - Sends the packet.

Syntax of the “no” Form

Traffic policing is removed by using the *no* form of this command:

no police

Defaults

- *burst-normal*: average rate multiplied by one second)
- *conform-action*: transmit
- *exceed-action*: drop
- *violate-action*: drop
- Command is disabled by default

Mode

Policy-Map Class configuration: **XSR(config-pmap-c-<*xx*>)#**

Example

The following example defines a traffic class using the **class-map** command and match criteria from the traffic class with the Traffic Policing configuration, which is configured in the service policy using the **policy-map** command. The **service-policy** command is then used to attach this service policy to the interface.

In this example, traffic policing is configured with the average rate of *8000* bits per second and the normal burst size at *1200* bytes and an excess burst of *2000* bytes for all packets leaving *F1/0*:

```
XSR(config)#class-map access-match
XSR(config-cmap<access-match>)#match access-group 1
XSR(config)#policy-map police-setting
XSR(config-pmap<police-setting>)#class access-match
XSR(config-pmap-c<access-match>)#police 8000 1200 2000 conform-action transmit
exceed-action drop
XSR(config>)interface fastethernet 1/0
XSR(config-if<F1>)#service-policy output police-setting
```

priority

This command gives priority to a class of traffic belonging to a policy map. It configures low latency queueing, providing strict Priority Queues (PQ) over Class-based Weighted Fair Queueing (CBWFQ). Strict PQ allows delay-sensitive data such as voice to be de-queued and sent before packets in other queues are dequeued.

The *burst* argument specifies the burst size and, as such, configures the network to accommodate temporary bursts of traffic. The default *burst* value, which is computed as 1 second of traffic at the configured *bandwidth* rate, is used when the *burst* argument is not specified.

Priority queues can be reserved by absolute bandwidth with these settings: *high, medium, low* and *normal*.

Note: The **bandwidth** and **priority** commands cannot be used in the same *class*, within the same policy map, but they can be used together in the same *policy map*. They cannot be configured for *class-default*. Class-default is always defined as fair queue.

Syntax

priority *priority-level bandwidth-kbps* [*burst*]

priority level	Specifies the priority queue: *high*, *medium*, *low* or *normal*. Normal priority has the least precedence.
bandwidth-kbps	Guaranteed allowed bandwidth for priority traffic. Beyond the guaranteed bandwidth, priority traffic will be dropped to ensure that non-priority traffic is not starved. Range: 1 to 100,000 kbps.
burst	Sets the burst size, ranging from 32 to 2,000,000 bytes.

Syntax of the “no” Form

Remove a previously specified priority specified for a class with the *no* form of this command:

```
no priority
```

Mode

Policy-Map Class configuration: **XSR(config-pmap-c-<*xx*>)#**

Example

The following example configures two PQs for the policy map *policy57*, with a high priority level, guaranteed bandwidth of *300* kbps and a one-time allowable burst size of *500* kbps for the map-class *voice;* and a low priority bandwidth, *80* bytes of guaranteed bandwidth, and a burst size *2000* bytes for map-class *beta*.

```
XSR(config)#policy-map policy57
XSR(config-pmap<policy57>)#class voice
XSR(config-pmap-c<voice>)#priority high 300 500
XSR(config-pmap<policy57>)#class beta
XSR(config-pmap-c<beta>)#priority low 80 2000
```

queue-limit

This command specifies or modifies the maximum number of packets the queue can hold for a class policy configured in a policy map.

Class-Based Weighted Fair Queueing (CBWFQ) creates a queue for every class for which a class map is defined. Packets satisfying the match criteria for a class accumulate in the queue reserved for the class until they are sent, which occurs when the queue is serviced by the Fair Queueing process. When the peak packet threshold you set for the class is reached, any further packet enqueueing to the class queue causes tail drop.

Syntax

queue-limit *number-of-packets*

number-of-packets	A number ranging from 1 to 64 specifying the peak number of packets that the queue can accomodate for this class.

Syntax of the “no” Form

The *no* form of the command removes the queue packet limit from a class. If RED is not configured, the queue limit is restored to the default value.

```
no queue-limit
```

Mode

Policy-Map Class configuration: **XSR(config-pmap-c-<*xx*>)#**

Default

64

Example

The following example configures policy map *policy75* to contain policy for class *acl203*. Policy for this class is set so that the queue reserved for it has a maximum packet limit of *50*.

```
XSR(config)#policy-map policy75
XSR(config-pmap<policy75>)#class acl203
XSR(config-pmap-c<acl203>)#bandwidth percent 35
XSR(config-pmap-c<acl203>)#queue-limit 50
```

random-detect (RED)

This command configures RED for a policy map.

This command configures and enables Random Early Detect (RED) for the class. RED is a congestion avoidance mechanism that slows traffic by randomly dropping packets during congestion and is useful with protocols like TCP that respond to dropped packets by reducing the transmission rate. While RED may be implemented using WRED, this command is retained for compatibility with earlier releases and simplicity of configuration when only RED is required.

Syntax

random-detect *min-thres max-thres* [*mark-prob*]

min-thres	Peak limit of average packet queue length, ranging from 1 to 4096, beyond which the XSR randomly drops packets.
max-thres	Peak limit of average packet queue length, ranging from 1 to 4096, beyond which all packets are dropped.
mark-prob	Mark probability denominator, ranging from 1 to 65,536. This is the liklihood of queued packets being dropped when their number exceeding the minimum threshold is between 0 and (1/*mark-prob*). When the peak threshold is reached, drop probability is 1 divided by the peak probability.

Syntax of the “no” Form

The *no* form of this command disable RED on an interface:

```
no random-detect
```

Mode

Policy-Map Class configuration: **XSR(config-pmap-c-<*xx*>)#**

Defaults

- Disabled
- Mark-prob: 10

Example

The following example enables RED. The minimum and maximum thresholds are *24* and *40*, respectively. The dropping probability is *1/4*.

```
XSR(config)#policy-map foobar
XSR(config-pmap<foobar>)#class alpha
XSR(config-pmap-c<alpha>)#random-detect 24 40 4
```

random-detect (WRED)

This command configures and enables Weighted Random Early Detect (WRED) for the class. WRED is a congestion avoidance mechanism that slows traffic by randomly dropping packets when congestion exists. WRED is useful with protocols like TCP that respond to dropped packets by decreasing the transmission rate.

To set or change WRED parameters, use the **random-detect** {*dscp* | *precedence*} command.

If no parameter passed to the command, the default is *prec-based* WRED.

Syntax

random-detect {*dscp-based* | *prec-based*}

dscp-based	WRED uses DSCP values when calculating drop probability.
prec-based	WRED uses IP precedence values when calculating drop probability.

Syntax of the "no" Form

The *no* form of this command disables WRED on an interface:

```
no random-detect
```

Mode

Policy-Map Class configuration: **XSR(config-pmap-c-<*xx*>)#**

Default

Prec-based

Example

The following example enables WRED as DSCP-based with the default values for parameters:

```
XSR(config)#policy-map DSCP
XSR(config-pmap<DSCP>)#class A
XSR(config-pmap-c<a>)#random-detect dscp-based
```

random-detect dscp

This command changes the Weighted Random Early Detect (WRED) minimum and maximum threshold and maximum drop probability for a DiffServ Code Point (DSCP) value.

This command specifies the DiffServ Code Point (DSCP) value. The DSCP can be a number from 0 to 63, or any of the following keywords: *af1, af12 , af13, af21, af22, af23, af31, af32, af33, af41, af42, af43, cs1, cs2, cs3, cs4, cs5, cs6, cs7, ef* or *default.* Each DSCP value has initial WRED parameters.

Table 12-1 provides initial parameter settings for each DSCP value. The last row details parameters for DSCP values not shown in the table.

Note: This command must be used in conjunction with the **random-detect** (interface) command. Also, **random-detect dscp** is available only if you specified the dscp-based argument when using the **random-detect** (interface) command.

Syntax

random-detect dscp *dscp-value min-thres max-thres* [*mark-prob*]

dscp-value	The DSCP value.
min-thres	Minimum limit of average packet queue length, ranging from 1 to 4096, beyond which the XSR randomly drops packets.
max-thres	Maximum limit of average packet queue length, ranging from 1 to 4096, beyond which all packets are dropped.
mark-prob	Mark probability denominator ranging from 1 to 65,536. This is the liklihood of queued packets being dropped when their number exceeding the minimum threshold is between 0 and (1/*mark-prob*). When the maximum threshold is reached, drop probability is 1 divided by the maximum probability.

Syntax of the "no" Form

The *no* form reverts WRED parameters to the default for a DSCP value:

no random-detect

Mode

Policy-Map Class configuration: **XSR(config-pmap-c-<*xx*>)#**

Defaults

- Disabled
- Default min-threshold settings used by the **random-detect dscp** command are shown in the following table. The *default max-threshold* and *mark-probability* are **40** and **1/10** respectively for all DSCP values.

Table 12-1 DSCP Threshold/Max Drop Probability Parameters

DSCP	Min Threshold	Max Threshold	Max Drop Probability
af11	32	40	1/10
af12	28	40	1/10
af13	24	40	1/10
Af21	32	40	1/10
Af22	28	40	1/10
Af23	24	40	1/10
Af41	32	40	1/10
Af31	28	40	1/10
Af32	24	40	1/10
Af33	32	40	1/10
Af42	28	40	1/10
Af43	24	40	1/10

Table 12-1 DSCP Threshold/Max Drop Probability Parameters (continued)

DSCP	Min Threshold	Max Threshold	Max Drop Probability
Cs1	32	40	1/10
Cs2	28	40	1/10
Cs3	24	40	1/10
Cs4	32	40	1/10
Cs5	28	40	1/10
Cs6	24	40	1/10
Cs7	32	40	1/10
Ef	28	40	1/10
Initial parameters for all other DSCP values	24	40	1/10

Examples

The following example enables WRED with a minimum threshold for DSCP *af21* of *24* and maximum threshold of *40*. The dropping probability is *1/4th*. All other DSCPs have default values.

```
XSR(config)#policy-map wred
XSR(config-pmap<wred>)#class a
XSR(config-pmap-c<a>)#random-detect dscp-based
XSR(config-pmap-c<a>)#random-detect dscp af21 24 40 4
```

The following example sets WRED It sets DSCP *33* WRED parameters to *10, 20, 10* and changes the setting for all other DSCP values from initial to default *5, 10, 20.*

```
XSR(config)#policy-map wred
XSR(config-pmap<wred>)#class a
XSR(config-pmap-c<a>)#random-detect dscp-based
XSR(config-pmap-c<a>)#random-detect dscp 33 10 20 10
XSR(config-pmap-c<a>)#random-detect default 5 10 20
```

random-detect exponential-weighting-constant

This command configures the Weighted Random Early Detect (WRED) exponential weight factor for the average queue size calculation. The weight constant is expressed as a power of 2.

WRED uses the exponential weighting factor to calculate average queue size. To simplify computing average queue size, the weight constant is allowed to be a power of 2. Choosing the right value of this constant is important for proper WRED operation. The default value is based on available data and should be changed only if your applications benefit from a different value.

Syntax

random-detect exponential-weighting-constant *value*

`value`	Exponent ranging from 1 to 16.

Syntax of the “no” Form

The *no* form of this command sets the constant to the default value of 9:

no random-detect exponential-weighting-constant

Mode

Policy-Map Class configuration: **XSR(config-pmap-c-<*xx*>)#**

Example

The following example enables *WRED* and sets the weight constant to *(1/2)^5*:

```
XSR(config)#policy-map wred
XSR(config-pmap<wred>)#class a
XSR(config-pmap-c<a>)#random-detect dscp-based
XSR(config-pmap-c<a>)#random-detect exponential-weighting-constant 5
```

random-detect precedence

This command sets Weighted Random Early Detect (WRED) the minimum and maximum threshold and maximum drop probability values for a IP precedence value.

The default WRED maximum drop probability (*MaxP*) is 1/10 and the default maximum threshold (*MaxTh*) is 40 for all IP precedence values. The default minimum threshold is calculated from *MaxTh* based on following formula:

MinTh = (1/2 - precvalue/16) x MaxTh

To change the default setting, use the **random-detect precedence default** command. By doing so, all IP precedence will share the same values except those which were explicitly configured with **random-detect precedence**. This setting is useful if WRED should operate as RED. To revert to the original default setting, enter **no random-detect precedence default**.

Syntax

random-detect precedence *prec-value min-thres max-thres* [*mark-prob*]default

prec-value	Precedence value, ranging from 0 to 7 with the keyword **default.**
min-thres	Minimum number of packets in the queue, ranging from 1 to 4096, beyond which the XSR randomly drops packets.
max-thres	Maximum number of packets in the queue, ranging from 1 to 4096, beyond which the XSR drops all packets.
mark-prob	Mark probability denominator. Liklihood of queued packets to be dropped when their number exceeding the minimum limit is between 0 and (1/mark-prob). Range: 1 to 65,536.

Syntax of the “no” Form

The *no* form of this command reverts WRED parameters to the default for a precedence value:

no random-detect precedence *prec-value*

Defaults

- Disabled
- Mark-prob: 10

Mode

Policy-Map Class configuration: **XSR(config-pmap-c-<*xx*>)#**

Examples

The following example enables *WRED* with a minimum IP precedence threshold of *24* and maximum of *40*. The dropping probability is *1/4*. All other precedence types have default values.

```
XSR(config)#policy-map wred
XSR(config-pmap<wred>)#class a
XSR(config-pmap-c<a>)#random-detect prec-based
XSR(config-pmap-c<a>)#random-detect precedence 3 24 40 4
```

The following example sets WRED as RED with a minimum threshold of *10* and maximum threshold of *20*:

```
XSR(config)#policy-map foo
XSR(config-pmap<foo>)#class a
XSR(config-pmap-c<a>)#random-detect prec-based
XSR(config-pmap-c<a>)#random-detect precedence default 10 20
```

set cos

This command marks the IEEE 802.1 priority in the header of output VLAN packets with a Class of Service (CoS) matching clause. As part of CoS configuration, the XSR associates a policy map with a class of traffic. By comparison, the **match cos** command marks the headers of *incoming* VLAN packets.

Note: Setting a VLAN priority value is applicable only to VLAN *sub-interfaces;* the set clause is ignored for other interface types.

For information on the **vlan** command, go to page 4-91 in the *Configuring Hardware Controllers* chapter.

Syntax

set cos *ieee802.1p-value*

ieee802.1p-value	Priority value to mark output VLAN packets, ranging from 0 to 7.

Syntax of the “no” Form

The *no* form of this command removes the match clause.

no set cos *ieee802.1p-value*

Mode

Policy-Map Class configuration: **`XSR(config-pmap-c-<xx>)#`**

Example

The following example configures *policy-map setCosTo4* that matches input priority value range from *5 to 7* and sets the output VLAN priority to *4*:

```
XSR(config)#policy-map setCosTo4
XSR(config-pmap<setCosTo4>)#class matchCos5To7
XSR(config-pmap-c<matchCos5to7>)#set cos 4
```

set ip dscp

This command marks a packet by setting the IP Differentiated Services Code Point (DSCP) in the Type of Service (ToS) byte. Once the IP DSCP bit is set, other QoS services can then operate on the bit settings.

Note: You cannot mark a packet by the IP precedence with the **`set ip precedence`** command and mark the same packet with an IP DSCP value by entering the **`set ip dscp`** command.

The network gives priority (or some type of expedited handling) to marked traffic. Typically, you set IP precedence at the edge of the network (or administrative domain); data then is queued based on the precedence. Class-Based Weighted Fair Queueing (CBWFQ) can speed up handling for high-precedence traffic at congestion points.

Note: Reserved keywords **EF** (Expedited Forwarding), **AF11** (Assured Forwarding Class 11), and **AF12** (Assured Forwarding Class 12) can be specified instead of numeric values.

Syntax

`set ip dscp` *`ip-dscp-value`*

`ip-dscp-value`	A number from 0 to 63 that sets the IP DSCP value. Reserved keywords can be set instead of numeric values as follows:
	af11 - Match packets with AF11 DSCP (001010)
	af12 - Match packets with AF12 DSCP (001100)
	af13 - Match packets with AF13 DSCP (001110)
	af21 - Match packets with AF21 DSCP (010010)
	af22 - Match packets with AF22 DSCP (010100)
	af23 - Match packets with AF23 DSCP (010110)
	af31 - Match packets with AF31 DSCP (011010)
	af32 - Match packets with AF32 DSCP (011100)
	af33 - Match packets with AF33 DSCP (011110)
	af41 - Match packets with AF41 DSCP (100010)
	af42 - Match packets with AF42 DSCP (100100)
	af43 - Match packets with AF43 DSCP (001010)

	cs1 - Match packets with CS1 DSCP (001000)
	cs2 - Match packets with CS2 DSCP (010000)
	cs3 - Match packets with CS3 DSCP (011000)
	cs4 - Match packets with CS4 DSCP (100000)
	cs5 - Match packets with CS5 DSCP (101000)
	cs6 - Match packets with CS6 DSCP (110000)
	cs7 - Match packets with CS7 DSCP (111000)
	default - Match packets with default DSCP (000000)
	ef - Match packets with Expedited Forwarding (EF) DSCP (101110)

Syntax of the "no" Form

The *no* form of this command removes a previously set IP DSCP:

```
no set ip dscp
```

Mode

Policy-Map Class configuration: **XSR(config-pmap-c-*xx*)#**

Example

In the following example, the IP DSCP TOS byte is set to *8* for *class1* and *cs2* for *class2* in *policy57*:

```
XSR(config)#policy-map policy57
XSR(config-pmap<policy57>)#class class1
XSR(config-pmap-c<class1>)#set ip dscp 8
XSR(config-pmap<policy57>)#class class2
XSR(config-pmap-c<class1>)#set ip dscp cs2
```

set ip precedence

This command sets the precedence value in the IP header. The network gives priority (or some type of expedited handling) to marked traffic through the application of CBWFQ or RED at points downstream in the network. Typically, you set IP Precedence at the edge of the network (or administrative domain); data then is queued based on the precedence. CBWFQ can speed up handling for certain precedence traffic at congestion points.

Syntax

set ip precedence *ip-precedence-value*

ip-precedence-value	Number from 0 to 7 that sets the precedence bit in the IP header.

Syntax of the "no" Form

The *no* form leaves the precedence value at its current setting:

```
no set ip precedence
```

Mode

Policy-Map Class configuration: **XSR(config-pmap-c-*xx*)#**

Example

The following example sets the IP Precedence bit to *7* for packets that satisfy the match criteria of the class map called *class39*. All packets that satisfy the match criteria of *class39* are marked with the IP Precedence value of *7*. How packets marked with the IP Precedence value of *7* are treated is determined by your network configuration.

```
XSR(config)#policy-map policy57
XSR(config-pmap<policy57>)#class class39
XSR(config-pmap-c<class39>)#set ip precedence 7
```

shape

This command enables and configures traffic shaping on a class. It can be applied to any fair-class or priority class. The default burst is sufficient to achieve the average rate and is calculated from the rate and the default measurement interval of 10 milliseconds:

Burst equals rate multiplied by (10 milliseconds divided by 1000)

In order to sustain the average rate, the normal burst cannot be less than the default burst. The default value for exceed burst is equal to the normal burst.

Syntax

shape rate [[*burst*]*exceed-burst*]

rate	Average or peak rate for output traffic in bbps.
burst	Maximum threshold burst size. Range: 1 to 20,000 bytes.
exceed-burst	Maximum exceed burst size. Range 1 to 40,000 bytes.

Syntax of the “no” Form

The *no* form of this command disables traffic shaping on a class:

```
no shape
```

Default

Disabled

Mode

Policy-Map Class configuration: **XSR(config-pmap-c-*xx*)#**

Example

The following example configures Class A with *20%* of the link bandwidth to a maximum of *64* Kbytes and maximum burst of *2000* bytes:

```
XSR(config)#policy-map foo
XSR(config-pmap<foo>)#class A
XSR(config-pmap-c<a>)#bandwidth percent 20
XSR(config-pmap-c<a>)#shape 64000 2000
```

Class-map Commands

class-map

This command creates a class map for matching packets to a specified class. Use it to specify the name of the class for which you want to create or modify class map match criteria.

Packets arriving at the output interface are checked against the match criteria set for a class map to determine if the packet belongs to that class. Sub-commands associated with the command are:

- **`match access-group`** - configures the match criteria for a class map on the basis of a configured ACL. Go to page 12-102 for the command definition.
- **`match cos`** - identifies a specific IEEE 802.1 priority value as a match criterion. Go to page 12-103 for the command definition.
- **`match ip dscp`** - identifies a specific IP Differentiated Service Code Point (DSCP) value as a match criterion. Go to page 12-103 for the command definition.
- **`match ip precedence`** - identifies IP precedence values as match criteria. Go to page 12-104 for the command definition.

Syntax

`class-map` {*match-all match-any*} *class-map-name*

match-all	Packets must match all criteria in the class-map to belong to the class-name.
match-any	Packets must match any (one or more) criteria in the class map to belong to the class-name.
class-map-name	Designation for the class-map which is used for the class map and to set policy for the class in the policy map.

Syntax of the “no” Form

Use the *no* form of this command to remove an existing class map:

`no class-map` [*match-all*] | [*match-any*] *word*

Mode

Global configuration: **`XSR(config)#`**

Next Mode

Class-Map configuration: **`XSR(config-cmap<xx>)#`**

Default

match-all

Example

The following example creates class-map *class57* and defines its match criterion with policy map *policy99* which is configured to contain policy rules for *class57* and the *default* class.

```
XSR(config)#class-map class57
XSR(config-cmap<class57>)#match access-group 136
XSR(config)#policy-map policy99
XSR(config-pmap<policy99>)#class class57
XSR(config-pmap-c<class57>)#bandwidth percent 10
XSR(config-pmap-c<class57>)#queue-limit 40
XSR(config-pmap<policy99>)#class class-default
XSR(config)#interface serial 1/0
XSR(config-if<S1/0>)#service-policy output policy99
```

match access-group

This command configures the match criteria for a class map on the basis of the specified Access Control List (ACL).

You define traffic classes based on match criteria including ACLs, DSCP and/or IP Precedence. Packets satisfying the match criteria for a class constitute the traffic for that class.

The **match access-group** command specifies a numbered ACL whose contents are used as the match criteria against which packets are checked to determine if they belong to the class set by the class map.

To use the **match access-group** command, you must first enter the **class-map** command to specify the name of the class whose match criteria you want to establish. After you identify the class, you can use one of the following commands to configure its match criteria:

- **match access-group**
- **match ip dscp**
- **match ip precedence**

Syntax

match access-group {*access-group*}

access-group	A numbered ACL whose contents are used as the match criteria against which packets are checked to determine if they belong to the class. Range: 1 to 199.

Syntax of the “no” Form

The *no* form of this command removes ACL match criteria from a class map:

no match access-group *access-group*

Mode

Class-map configuration: **XSR(config-cmap-*xx*)#**

Example

The following example specifies a class map called *acl57* and configures the ACL numbered *57* to be used as the match criteria for this class:

```
XSR(config)#class-map acl57
XSR(config-cmap<acl57>)#match access-group 57
```

match cos

This command identifies a specific IEEE 802.1 priority value as a match criterion. Up to 8 priority values can be matched in one match statement. For example, if you wanted the priority values of 0, 1, 2, 3, 4, 5, 6, or 7 (note that only one of the priority values must be a successful match criterion, not all of the specified priority values), enter the **match cos 0 1 2 3 4 5 6 7** command.

This command is used by the class map to identify a specific priority value marking on the header of incoming VLAN packets. By comparison, the **set cos** command marks the headers of *outgoing* VLAN packets. For information on the **vlan** command, go to page 4-91 in the *Configuring Hardware Controllers* chapter.

Syntax

match cos ieee802.1p-value [*ieee802.1p-value*] [*ieee802.1p-value*] ...

ieee802.1p-value	Priority value in the input VLAN header, ranging from *0* to *7*.

Syntax of the "no" Form

The *no* form of this command removes the match clause:

no match cos

Default

No match clause for VLAN priority

Mode

Class-map configuration: **XSR(config-cmap-*xx*)#**

Example

The following example example configures classmap *matchCos5To7* that matches input priority values from *5 to 7*:

```
XSR(config)#class-map matchCos5To7
XSR(config-cmap<matchCos5To7>)#match cos 5 6 7
```

match ip dscp

This command identifies a specific IP Differentiated Service Code Point (DSCP) value as a match criterion. Up to 8 IP DSCP values can be matched in one match statement. For example, if you wanted the IP DCSP values of 0, 1, 2, 3, 4, 5, 6, or 7 (note that only one of the IP DSCP values must be a successful match criterion, not all of the specified IP DSCP values), enter the match **ip dscp 0 1 2 3 4 5 6 7** command.

This command is used by the class map to identify a specific IP DSCP value marking on a packet. The *ip-dscp-value* arguments are used as markings only. The IP DSCP values have no mathematical significance. For instance, the *ip-dscp-value* of 2 is not greater than 1. The value simply indicates that a packet marked with the *ip-dscp-value* of 2 is different than a packet marked with the *ip-dscp-value* of 1. The treatment of these marked packets is defined by the user through the setting of QoS policies in policy-map class configuration mode.

Syntax

match ip dscp *ip-dscp-value [ip-dscp-value][ip-dscp-value] [ip-dscp-value] [ip-dscp-value][ip-dscp-value][ip-dscp-value][ip-dscp-value]*

ip-dscp-value	Specifies a value from 0 to 63 to identify an IP DSCP value.

Syntax of the "no" Form

To remove a specific IP DSCP value from a class map, use the *no* form of this command:

no match ip dscp *ip-dscp-value [ip-dscp-value][ip-dscp-value][ip-dscp-value][ip-dscp-value][ip-dscp-value][ip-dscp-value][ip-dscp-value]*

Mode

Class-map configuration: **XSR(config-cmap-*xx*)#**

Example

The following example shows how to configure the service policy called *priority55* and attach service policy *priority55* to an interface. In this example, the class map *ipdscp15* will evaluate all packets entering interface *F1* for an IP DSCP value of *15*. If the incoming packet has been marked with the IP DSCP value of 15, the packet will be treated with a high priority level.

```
XSR(config)#class-map ipdscp15
XSR(config-cmap<ipdscp15>)#match ip dscp 15
XSR(config)#policy-map priority55
XSR(config-pmap<priority55>)#class ipdscp15
XSR(config-pmap-c<ipdscp15>)#priority high 55
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#service-policy output priority55
```

match ip precedence

This command identifies IP precedence values as match criteria. Up to 4 precedence values can be matched in one match statement. For example, if you wanted the IP precedence values of 0, 1, 2, or 3 (note that only one of the IP precedence values must be a successful match criterion, not all of the specified IP precedence values), enter the **match ip precedence 0 1 2 3** command.

The *ip-precedence-value* arguments are used as markings only, they have no mathematical significance. For instance, the ip-precedence-value of 2 is not greater than 1. The value simply indicates that a packet marked with the ip-precedence-value of 2 is different than a packet marked with the ip-precedence-value of 1. You define the treatment of these different packets by setting QoS policies in Policy-map Class configuration mode.

Syntax

match ip precedence *ip-precedence-value [ip-precedence-value] [ip-precedence-value][ip-precedence-value][ip-precedence-value][ip-precedence-value][ip-precedence-value] [ip-precedence-value][ip-precedence-value]*

ip-precedence-value	Specifies an IP precedence value from 0 to 7.

Syntax of the "no" Form

Use the *no* form of this command to remove IP precedence values from a class map:

no match ip precedence *ip-precedence-value* [*ip-precedence-value*] [*ip-precedence-value*][*ip-precedence-value*][*ip-precedence-value*][*ip-precedence-value*][*ip-precedence-value*] [*ip-precedence-value*][*ip-precedence-value*]

Mode

Class-map configuration: **XSR(config-cmap-*xx*)#**

Example

The following example shows how to configure the service policy called *priority50* and attach service policy *priority50* to an interface. In this example, the class map called *ipprec5* will evaluate all packets entering *F1/0/0* for an IP precedence value of 5. If the incoming packet has been marked with the IP precedence value of *5*, the packet will be treated with a priority level of *50*.

```
XSR(config)#class-map ipprec5
XSR(config-cmap<ipprec5>)#match ip precedence 5
XSR(config)#policy-map priority50
XSR(config-pmap<priority50>)#class ipprec5
XSR(config-pmap-c<ipprec5>)#priority high 50
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#service-policy output priority50
```

QoS Show Commands

show class-map

This command displays all class maps and their matching criteria.

You can use the **show class-map** command to display all class maps and their matching criteria. If you enter the optional *class-map-name* argument, the specified class map and its matching criteria will be displayed.

Syntax

show class-map [*class-map-name*]

class-map-name	Name of the class map.

Mode

EXEC, Privileged EXEC, or Global configuration: **XSR>**, **XSR#**, or **XSR(config)#**

Sample Output

In this example, three class maps are defined. Packets that match access list *103* belong to class *c3*, IP packets with IP precedence belong to class *c2*, and packets with *DSCP 32* belong to class *c1*. The output from the **show class-map** command shows the three defined class maps.

```
XSR#show class-map
```

```
Class map c3
Match access-group 103
Class map c2
Match ip precedence 2
Class map c1
Match ip dscp 32
```

show policy-map

This command displays the configuration of all classes for a specified service policy map or all classes for all existing policy maps. It displays the configuration of a service policy map created using the **policy-map** command. You can use the **show policy-map** command to display all class configurations comprising any existing service policy map, whether or not that service policy map has been attached to an interface.

Syntax

show policy-map [*policy-map*] *interface-type*

policy-map	Service policy map name whose complete configuration will be shown.
interface type	Configuration for classes on the specified interface including: *ATM, BRI, Fast/GigabitEthernet, Loopback, Serial, Multilink, or Dialer (0* to *255).*

Default

All existing policy map configurations are displayed.

Mode

EXEC, Privileged EXEC, or Global configuration: **XSR>, XSR#,** or **XSR(config)#**

Sample Output

This example displays the contents of the service policy map called *po1*:

```
XSR#show policy-map po1
Policy Map po1
 CLass c1: Weighted Fair Queue bandwidth 600 (kbps)
 Class c2: Weighted Fair Queue bandwidth 300 (kbps)
```

This example displays the contents of all policy maps on the XSR:

```
XSR#show policy-map
Policy Map p6
 Class c1: Weighted Fair Queue bandwidth 10 %
 Class c2: Weighted Fair Queue bandwidth 80 %

Policy Map p9
 Class c1: Priority high bandwidth 300 (kbps)
 Class c2: Weighted Fair Queue bandwidth 800 (kbps)

Policy Map p10
 Class c1: Weighted Fair Queue bandwidth 600 (kbps)
 Class c2: Weighted Fair Queue bandwidth 300 (kbps)
```

show policy-map interface

This command shows the configuration of all service policies applied on an interface or Frame Relay Data-link Connection Identifier (DLCI). It displays the configuration for classes on the specified interface or specified DLCI only if a service policy has been attached to the interface or PVC. This command shows input and the output policies applied to the interfaces.Counters displayed after you enter the **`show policy-map`** interface command are updated only if congestion is present on the interface.

Note: This command displays policy information about Frame Relay PVCs only if Frame Relay Traffic Shaping (FRTS) is enabled on the interface.

Counters displayed after you enter the show policy-map interface command are updated only if congestion is present on the interface.

When QoS is applied to a Dialer interface, this command displays no data. To display the policy-map after the dialer has built the connection, enter the show policy map command on the interface from the dialer pool that the dialer called on and not the dialer interface itself.

Syntax

`show policy-map interface` *`interface-type`* [**`dlci`** *`dlci`*] *`mlpppgroup`*

`interface type`	Interface or sub-interface type including: *ATM, BRI, Fast/GigabitEthernet, Loopback, Multilink,* or *Dialer (0-255).*
`dlci`	A specific PVC for which policy configuration is shown.
`dlci`	A Data-Link Connection Identifier (DLCI) number used on the interface. Policy configuration for the corresponding PVC is shown when a DLCI is specified.
`mlpppgroup`	Multilink PPP group number.

Mode

Privileged EXEC or Global configuration: **`XSR#`** or **`XSR(config)#`**

Sample Output

The following example shows policy map *mypolicy* attached to DLCI *100* on Serial interface 1/0. Policy is applied simultaneously to input and output traffic. Input policy displays counters for input QoS (actual bandwidth and policing). Shaping, bandwidth and buffer management are not perfomred on input traffic and are shown for output traffic only.

```
XSR(config)#policy-map mypolicy
XSR(config-pmap<mypolicy>)#exit
XSR(config)#class-map smallPackets
XSR(config-pmap-c<smallPackets>)#priority high 800
XSR(config-pmap-c<smallPackets>)#random-detect 20 25 2
XSR(config-pmap-c<smallPackets>)#class immediate-data
XSR(config-pmap-c<immediate-data>)#bandwidth 300
XSR(config-pmap-c<immediate-data>)#class class-default
XSR(config-pmap-c<class-default>)#shape 100000 12500
```

```
XSR(config)#map-class frame-relay foo
XSR(config-map-class<foo>)#frame-relay cir out 100000
XSR(config-map-class<foo>)#frame-relay bc out 10000
XSR(config-map-class<foo>)#service-policy output mypolicy
XSR(config-map-class<foo>)#service policy input mypolicy

XSR#show policy-map interface s1/0.1 dlci 100

Serial1/0.1: DLCI 100 -
output: mypolicy
Class smallPackets
 Priority High
   Bandwidth 800 (kbps)Actual bandwidth 0 (kbps),
   Random-detect :
   Avg Qsize: 5.32, Random Drops : 54
   min-th : 20 max-th : 25 mark-prob : 1/2
   Tx/NoBuff/Error (19892/35/0)

Class immediate-data
  Weighted Fair Queuing
   Bandwidth 300 (kbps) Actual bandwidth 0 (kbps),
        Max Qsize: 64, Qsize: 32, Tail drops 223
        Tx/NoBuff/Error (3321/22/0)

Class class-default
Weighted Fair Queuing
   Bandwidth 436 (kbps) Actual bandwidth 0 (kbps),
   Max Qsize: 64, Qsize: 0, Tail drops 0
   Tx/NoBuff/Error (0/0/0)
   Traffic shaping
   Average Normal  Exceed  Refresh Refresh
   Rate    Burst   Burst   Time    Bytes
   100000  12500   0       10(ms)  125

Serial1/0.1: DLCI 100 -
input : mypolicy
Class smallPackets
 Actual bandwidth 12 (kbps) Tx/NoBuff/Error (19892/0/0)
Class immediate-data
 Actual bandwidth 0 (kbps) Tx/NoBuff/Error (3321/0/0)
Class class-default
Actual bandwidth 0 (kbps)Tx/NoBuff/Error (0/0/0)
```

Parmeter Descriptions

Bandwidth	Configured bandwidth for a class in percentage or kbps.
Actual bandwidth	Bandwidth that this class actually receives on the output link.
Max Qsize	Configured queue size.
Qsize	Current queue size.

Tail drops	Sum of packets dropped by Tail Drop buffer management.
Tx	Sum of packets transmitted successfully.
NoBuff	Sum of packets rejected by the driver because of no buffer. This value is always zero when the policy map is applied to DLCI and MLPPP.
Error	Sum of transmit (driver) errors when trying to send out a packet. Value is always zero when the policy map is applied to DLCI and MLPPP.
Avg Qsize	RED average queue size.
Random Drops	Sum of packets dropped by RED.
min-th	Configured minimum threshold for RED.
max-th	Configured maximum threshold for RED.
mark-prob	Configured mark probability for RED.

show random-detect interface

This command displays data about Random Early Detection (RED).

Syntax

show random-detect interface [*interface-type interface-number*]

interface-type	The type of interface.
interface-number	The number of the interface.

Mode

EXEC: **XSR>** or **XSR(config)#**

Sample Output

The following commands configure policy-map *Shape*:

```
XSR(config)#policy-map Shape
XSR(config-pmap<Shape>)#class d32
XSR(config-pmap-c<d32>)#bandwidth per 30
XSR(config-pmap-c<d32>)#random-detect dscp-based
XSR(config-pmap-c<d32>)#random-detect dscp 32 10 20 10
XSR(config-pmap-c<d32>)#random-detect dscp default 2 5 20
```

The following is sample output from the command. There are drops only from class *d32*.

```
XSR#show random-detect interface serial 1/0:0

Serial 1/0:0

output: Shape

output: Shape
Class d32
        Weighted Random-detect:
        Avg Qsize: 5, Total Random Drops: 2223
```

DSCP	min-th	max-th	mark-prob	tail drop	early drop
0	2	5	20	0	0
1	2	5	20	0	0
2	2	5	20	0	0
3	2	5	20	0	0
4	2	5	20	0	0
5	2	5	20	0	0
6	2	5	20	0	0
7	2	5	20	0	0
8	2	5	20	0	0
9	2	5	20	0	0
10	2	5	20	0	0
11	2	5	20	0	0
12	2	5	20	0	0
13	2	5	20	0	0
14	2	5	20	0	0
15	2	5	20	0	0
16	2	5	20	0	0
17	2	5	20	0	0
18	2	5	20	0	0
19	2	5	20	0	0
20	2	5	20	0	0
21	2	5	20	0	0
22	2	5	20	0	0
23	2	5	20	0	0
24	2	5	20	0	0
25	2	5	20	0	0
26	2	5	20	0	0
27	2	5	20	0	0
28	2	5	20	0	0
29	2	5	20	0	0
30	2	5	20	0	0
31	2	5	20	0	0
32	2	5	20	1900	323
33	10	20	10	0	0
34	2	5	20	0	0
35	2	5	20	0	0
36	2	5	20	0	0
37	2	5	20	0	0
38	2	5	20	0	0
39	2	5	20	0	0
40	2	5	20	0	0
41	2	5	20	0	0
42	2	5	20	0	0
43	2	5	20	0	0
44	2	5	20	0	0
45	2	5	20	0	0
46	2	5	20	0	0
47	2	5	20	0	0
48	2	5	20	0	0

```
49      2       5       20              0       0
50      2       5       20              0       0
51      2       5       20              0       0
52      2       5       20              0       0
53      2       5       20              0       0
54      2       5       20              0       0
55      2       5       20              0       0
56      2       5       20              0       0
57      2       5       20              0       0
58      2       5       20              0       0
59      2       5       20              0       0
60      2       5       20              0       0
61      2       5       20              0       0
62      2       5       20              0       0
63      2       5       20              0       0
        Exponential weighting constant: 9
```

Parameter Descriptions

Average Queue size	Average output queue size for this interface.
Total Random Drops	Sum of packets dropped for all DSCP codepoint..
Min-th	Minimum threshold.
Max-th	Maximum length of the queue. When the average queue size is larger than this number, any additional packets will be dropped.
Mark-prob	Probability (1/mark-prob) for random drops.
DSCP	DSCP code point.
Tail drop	Number of drops because of average queue size greater than max-threshold.
Early drop	Number of drops when the average queue size is between min-threshold and max-threshold.

show shape interface

This command displays information about QoS traffic shaping.

Syntax

show shape interface [*interface-type interface-number*]

interface-type	Type of interface.
interface-number	Number of the interface.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

This following commands configure shape information for each class. In the following example policy-map *shape* is configured as follows:

```
XSR(config)#policy-map Shape
XSR(config-pmap<Shape>)#class d32
XSR(config-pmap-c<d32>)#bandwidth per 30
XSR(config-pmap-c<d32>)#shape 400000 50000
XSR(config-pmap-c<d32>)#class d33
XSR(config-pmap-c<d33>)#bandwidth per 30
XSR(config-pmap-c<d32>)#shape 100000 12500
```

The following is sample output displays shape information for classes *d32* and *d33*:

```
XSR# show shape interface serial 1/0:0

Serial 0/1/0:0
output: Shape
Serial 0/1/1:1
output: Shape
Class d32
        Traffic shaping
        Average Normal  Exceed  Refresh Refresh
        Rate    Burst   Burst   Time    Bytes
        400000  50000   0       10(ms)  500
Class d33
        Traffic shaping
        Average Normal  Exceed  Refresh Refresh
        Rate    Burst   Burst   Time    Bytes
        100000  12500   0       10(ms)  125
```

Parameter Descriptions

Average Rate	Average shaped rate configured.
Normal burst	Configured normal burst.
Exceed burst	Configured exceed burst.
Refresh time	Time interval of bucket refill with tokens.
Refresh bytes	Number of bytes added to the bucket per time interval.

13

Configuring ADSL

Observing Syntax and Conventions

The CLI command syntax and conventions use the notation described below.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[x]	[ ] Square brackets indicate an optional parameter (*italic*)
[x | y | z]	[|] Square brackets with vertical bar indicate a choice of values
{x | y | z}	{ | } Braces with vertical bar indicate a choice of a required value
[x {y | z}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<*xx*>)	*xx* signifies interface type and number, e.g.: **F1, S2/1.0, D1, M57, L1, ATM0/1/1**
Next Mode entries display the CLI prompt after a command is entered.	
Sub-command headings are displayed in ***red***, italicized text.	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

ADSL Configuration Commands

The following command sets define ADSL functionality on the XSR including:

- "CMV Commands" on page 13-83.
- "Other ADSL Commands" on page 13-87.
- "PPP Configuration Commands" on page 13-99.
- "ATM Clear and Show Commands" on page 13-103.

CMV Commands

cmv append

This command adds a Command Management Variable (CMV) to the DSP training list which is used by the DSP firmware when the line is in training mode. This command is intended for use by Enterasys field service personnel only. This command requires that the ADSL NIM be installed and the DSP firmware file be present in the **Flash:** directory.

Syntax

cmv append command-ID offset value

command-ID	Represents a 4-character CMV command.
offset	Decimal or hexadecimal number representing where to write the value.
value	Decimal or hexadecimal number.

Mode

ATM Interface configuration: **XSR(config-if<ATMxx>)#**

Example

The following example adds the CMV *DOPT 1* with a hex value:

```
XSR(config-if<ATM0/1/1>)#cmv append DOPT 1 0x306090c0
```

cmv clear

This command removes all Command Management Variable (CMV) commands from the CMV training list which is used by the DSP firmware when the line is in training mode. This command is intended for use by Enterasys field service personnel only.

This command requires that the ADSL NIM be installed and the DSP firmware file be present in the **Flash:** directory.

Syntax

cmv clear

Mode

ATM Interface configuration: **XSR(config-if<ATMxx>)#**

Example

The following example deletes all CMVs from the training list:

```
XSR(config-if<ATM0/1/1>)#cmv clear
```

cmv cr

This command *reads* a Command Management Variable (CMV) from the DSP. This command is intended for use by Enterasys field service personnel only.

This command requires that the ADSL NIM be installed and the DSP firmware file be present in the **Flash:** directory.

Syntax

cmv cr command-ID offset

command-ID	Represents a 4-character CMV command.
offset	Decimal or hexadecimal number representing where to read the value.

Mode

ATM Interface configuration: **XSR(config-if<ATMxx>)#**

Example

The following example reads CMV *STAT 0* from the DSP:

```
XSR(config-if<ATM0/1/1>)#cmv cr STAT 0
```

cmv cw

This command *writes* a Command Management Variable (CMV) to the DSP. This command is intended for use by Enterasys field service personnel only.

This command requires that the ADSL NIM be installed and the DSP firmware file be present in the **Flash:** directory.

Syntax

cmv cw command-ID offset value

command-ID	Represents a 4-character CMV command.
offset	Decimal or hexadecimal number representing where to write the value.
value	Decimal or hexadecimal number.

Mode

ATM Interface configuration: **XSR(config-if<ATMxx>)#**

Example

The following example writes *UOPT 2* with a hex value to the DSP:

```
XSR(config-if<ATM0/1/1>)#cmv cw UOPT 2 0x0c0e1014
```

cmv delete

This command deletes the specified Command Management Variable (CMV) from the DSP retaining list which is used by the DSP firmware when the line is in training mode. This command is intended for use by Enterasys field service personnel only.

This command requires that the ADSL NIM be installed and the DSP firmware file be present in the **Flash:** directory.

Syntax

cmv delete command-ID offset [value]

command-ID	Represents a 4-character CMV command.
offset	Decimal or hexadecimal number representing where to write the value.
value	Decimal or hexadecimal number

Mode

ATM Interface configuration: **XSR(config-if<ATMxx>)#**

Example

The following example deletes CMV *OPTN2*, from the retaining list:

```
XSR(config-if<ATM0/1/1>)#cmv delete OPTN 2
```

cmv print

This command prints the Command Management Variable (CMV) training list on the console. The training list is used by the DSP firmware when the line is in training mode. This command is intended for use by Enterasys field service personnel only.

This command requires that the ADSL NIM be installed and the DSP firmware file be present in the **Flash:** directory.

Syntax

cmv print

Mode

ATM Interface configuration: **XSR(config-if<ATMxx>)#**

Example

The following example prints the CMV training list to the console:

```
XSR(config-if<ATM0/1/1>)#cmv print
```

cmv save

This command saves the Command Management Variable (CMV) training list to a file. The training list is used by the DSP firmware when the line is in training mode. This command is intended for use by Enterasys field service personnel only.

This command requires that the ADSL NIM be installed and the DSP firmware file be present in the **Flash:** directory.

Syntax

cmv save file-name

file-name	The name of the file used to save the CMV training list.

Mode

ATM Interface configuration: **XSR(config-if<ATMxx>)#**

Example

The following example saves the CMV training list to file *retrain-list*:

```
XSR(config-if<ATM0/1/1>)#cmv save retrain-list
Save complete
XSR(config-if<ATM0/1/1>)#
```

Other ADSL Commands

description

This command adds a description string to an existing ATM interface object.

This command requires that the ADSL NIM be installed and the DSP firmware file be present in the **Flash:** directory.

Syntax

description description_text

description_text	A text string that describes the interface object. Text with embedded spaces must be enclosed in double quotes. Omitting the description text results in an empty description string.

Syntax of the “no” Form

The *no* form of this command sets the description text to an empty string:

no description

Mode

ATM Interface configuration: **XSR(config-if<ATMxx>)#**

Example

The following example adds *ADSL Line* to the interface object:

```
XSR(config-if<ATM0/1/1>)#description "ADSL Line"
```

interface atm

This command creates an ATM interface object and its associated device driver which downloads the specified firmware file to the on-board DSP. Depending on the size of the DSP firmware and the characteristics of the download procedure, this procedure may take a noticeable amount of time. After a successful load, the interface and device driver is in the administrative down state (*shutdown*).

Caution: This command requires that the ADSL NIM be installed and the DSP firmware file be present in the **Flash:** directory.

Syntax

interface atm {*slot/card/port*}

slot	The XSR slot number, ranging from 0 to 2.
card	The XSR NIM number, ranging from 1 to 2.
port	The XSR slot number: 0. The sub-interface number ranges from 1 to 30.

Syntax of the “no” Form

The *no* form of this command removes the interface object and all associated sub-interface objects. The interface must be shut down first.

no interface atm {*slot/card/port*}

Mode

Global configuration: **XSR(config)#**

Next Mode

ATM Interface configuration: **XSR(config-if<ATMxx>)#**

Example

The following example creates an ATM interface on slot *0*, card *1*, port *1*:

```
XSR(config)#interface atm 0/1/1
XSR(config-if<ATM0/1/1>)#
```

interface atm sub-interface

This command creates an ATM sub-interface object and associates it with its ATM interface peer. Setup of internal data paths, which will route an IP interface to the ATM sub-interface, will continue as configuration proceeds and a **no shutdown** command has been issued against this sub-interface instance. On successful construction, the sub-interface is in the administrative down state (*shutdown*).

This command requires that the ADSL NIM be installed, the DSP firmware file be present in the **Flash:** directory, and the ATM port be properly configured.

The following commands are sub-commands of **atm sub-interface**:

- **backup** - configures and enables a backup interface for the ATM sub-interface. Refer to page 13-90 for the command description.
- **crypto** - enables and configures VPN parameters on the sub-interface. Refer to page 13-92 for the command description.
- **description** - adds a description string to an existing ATM sub-interface. Refer to page 13-92 for the command description.
- **encapsulation** - selects the data encapsulation method for this ATM sub-interface. Refer to page 13-92 for the command description.
- **exit** - quits ATM Sub-Interface mode and returns to Global mode. Refer to page 13-93 for the command description.
- **ip address** - specifies the IP address and subnet mask of the ATM sub-interface or requests the IP address and subnet mask be negotiated. Refer to page 13-93 for the command description.
- **no shutdown** - sets the ATM sub-interface to the administrative *up* state and enables the virtual circuit. Refer to page 13-94 for the command description.
- **oam-pvc** - enables end-to-end F5 (circuit) OAM cell procedures for ATM Permanent Virtual Circuit (PVC) management. Refer to page 13-95 for the command description.
- **oam-retry** - configures parameters related to OAM cell handling for ATM VC management. Refer to page 13-96 for the command description.
- **pvc** - sets the sub-interface circuit type to PVC and specifies ATM VPI/VCI values. Refer to page 13-96 for the command description.
- **shutdown** - sets the ATM sub-interface to the administrative *Down* state halting all data traffic on this VC. Refer to page 13-97 for the command description.

Syntax

interface atm {*slot/card/port.sub-interface*}[*point-to-point*]

slot	The XSR slot number, ranging from 0 to 2.
card	The XSR NIM number, ranging from 1 to 2.
port	The XSR slot number: 0.
sub-interface	Identifies a sub-interface on that interface, ranging from 1 to 30.
point-to-point	Interoperability option.

Syntax of the “no” Form

The *no* form of this command deletes the sub-interface object:

no interface atm [*slot/*]*card/port.sub-interface* [*point-to-point*]

Mode

Global configuration: **XSR(config)#**

Next Mode

ATM Sub-Interface configuration: **XSR(config-if<ATMxx.x>)#**

Defaults

- *Backup*: Disabled
- *VPN*: Disabled
- *Description*: Set to the empty string
- *Encapsulation*: None
- *IP*: Not configured
- *PPP*: Not configured
- *OAM procedures*: Disabled
- *ATM PVC VPI/VCI*: Set to 1/32
- The *sub-interface* will be in the shutdown state

Example

The following example creates an ATM sub-interface object on ATM interface slot *0*, card *1*, port *1*:

```
XSR(config)#interface atm 0/1/1.1 point-to-point
XSR(config-if<ATM0/1/1.1>)#
```

backup

This command configures and enables a backup interface for this ATM sub-interface. This command requires a properly configured ATM sub-interface and Dialer group.

Syntax

backup {**delay** *down-wait* {*up-wait* | *never*} | **interface dialer** *id* | **time-range** *begin-hh:mm end-hh:mm*}

down-wait	Seconds to wait before switching to the backup interface.
up-wait | **never**	Seconds to wait before switching back to ATM interface. If set to **never,** it will remain on the backup interface.
id	Dialer to use for backup when ATM interface is down.
begin-hh:mm	Time of day to switch to the backup line regardless of ATM interface state.
end-hh:mm	Time of day to revert to normal interface backup procedures.

Syntax of the “no” Form

The *no* form of this command disables a backup for this ATM sub-interface:

no backup {**delay** | **interface** | **time-range**}

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATM*x*/*x*/*x*.*x*>)#**

Default

Disabled by default. When enabled, all operational parameters must be specified.

Example

The following example configures a sub-interface backup with a Dialer ID of 1, delay of 20 seconds before switching to the backup, and a delay of 10 seconds before switching back to the ATM sub-interface. The example also configures the sub-interface to switch to the backup line at 8:30 P.M. then switch back to the normal interface at 9:50 P.M. :

```
XSR(config-if<ATM0/1/0.1>)#backup interface Dialer1
XSR(config-if<ATM0/1/0.1>)#backup delay 20 10
XSR(config-if<ATM0/1/0.1>)#backup time-range 20:30 21:50
```

crypto

This command enables and configures the DF-bit VPN parameter on this ATM sub-interface. This command requires a properly configured ATM sub-interface.

Syntax

crypto {*ezipsec* | **ipsec df-bit** {*clear* | *copy* | *set*}| **map** [*map-name*]}

ezipsec	EZ-IPSec automatic configuration enabled.
ipsec df-bit	IPSec enabled with the following DF-bit options:
clear	The outer IP header clears the DF bit and the XSR may fragment the packet to add IPSec encapsulation.
copy	XSR searches the original packet for the outer DF-bit setting.
set	The outer IP header has the DF-bit set; but, the XSR may fragment the packet if the original packet cleared the DF-bit.
map-name	Attaches a crypto map to the interface and name (optional).

Syntax of the “no” Form

This command’s *no* disables the specified DF-bit setting:

no crypto {*ezipsec* | **ipsec df-bit**}| **map** [*map-name*]}

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATM*x*/*x*/*x*.*x*>)#**

Default

Disabled

Example

The following example enables EZ-IPSec with the option of having the XSR look in the original packet for the outer DF bit setting. This example also attaches the crypto map *ets-vpn*:

```
XSR(config-if<ATM0/1/0.1>)#crypto ezipsec
XSR(config-if<ATM0/1/0.1>)#crypto ipsec df-bit copy
XSR(config-if<ATM0/1/0.1>)#crypto map ets-vpn
```

description

This command adds a description string to an existing ATM sub-interface. This command requires a properly configured ATM sub-interface.

Syntax

description description_text

description _text	A string describing the sub-interface object. Text with embedded spaces must be enclosed in double quotes. Omitting text causes an empty string.

Syntax of the "no" Form

The *no* form of this command sets the description text to an empty string:

no description

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATMxx.x>)#**

Example

The following example adds the *ADSL VC 1/32* text string to the sub-interface object:

XSR(config-if<ATM0/1/0.1>)#description "ADSL VC 1/32"

encapsulation

This command selects the data encapsulation method for this ATM sub-interface. Be aware that an encapsulation method must be selected before the sub-interface can pass data.

Note: This command requires a properly configured ATM sub-interface. In order to change encapsulation, you must issue the **no encapsulation** command first before restting the value.

Syntax

encapsulation {*mux* | *snap*}{*ipoa* | *pppoa* | *pppoe*} [*service-name*]

mux	VC multiplexing (per RFC-2684/1483).
snap	LLC/SNAP multiplexing (per RFC-2684/1483).
ipoa	IP encapsulated traffic flows on this VC (per RFC-2684).
pppoa	PPP encapsulated traffic flows on this VC (per RFC-2364).
pppoe	PPP over Ethernet encapsulated traffic flows on this VC (per RFC-2516).

service -name	The name of the PPPoE service. If not specified, PPPoE connects to the first advertised service name. At this time, the XSR will connect with the first advertised service name only.

Syntax of the "no" Form

The *no* form of this command removes any form of encapsulation, effectively disabling the sub-interface:

```
no encapsulation
```

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATM*xx.x*>)#**

Default

The default encapsulation is none. An encapsulation method must be specified before the sub-interface can pass data. When the sub-interface is configured for PPPoE encapsulation, the source Ethernet MAC address will be set to the MAC address of FastEthernet interface 2.

Example

The following example configures the sub-interface for LLC/SNAP multiplexing and PPPoA encapsulated traffic:

```
XSR(config-if<ATM0/1/0.1>)#encapsulation snap pppoa
```

exit

This command quits the ATM Sub-Interface mode and returns to Global mode.

Syntax

```
exit
```

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATM*xx.x*>)#**

Example

The following example exits the sub-interface ATM command mode from ATM interface slot *0*, card *1*, port *0*, sub-interface *1*:

```
XSR(config-if<ATM0/1/0.1>)#exit
XSR(config)#
```

ip address

This command specifies the IP address and subnet mask of the ATM sub-interface or requests the IP address and subnet mask be negotiated. This command requires a properly configured ATM sub-interface.

Syntax

ip address {*ip-address/subnet-mask* | *negotiated*}

ip-address	The IP address associated with this sub-interface in the form: A.B.C.D.
subnet-mask	The subnet mask bits represents the number of bits set to 1 in the subnet mask, ranging from 0 to 32.
negotiated	IP address/subnet mask are negotiated by PPP. This value cannot be set when using IPoA encapsulation.

Syntax of the "no" Form

The *no* form of this command returns this parameter to its default setting:

no ip address

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATM*xx.x*>)#**

Default

- IP address: 0.0.0.0
- Subnet mask: 0.0.0.0.

Example

This example sets the sub-interface IP address to *10.1.1.1* and the subnet mask to *255.0.0.0*:

```
XSR(config-if<ATM0/1/0.1>)#ip address 10.1.1.1 255.0.0.0
```
or
```
XSR(config-if<ATM0/1/0.1>)#ip address 10.1.1.1/8
```

no shutdown

This command sets the ATM sub-interface to the administrative *Up* state (**no shutdown**) and enables the virtual circuit.

The associated ATM interface must be in the administrative Up state (**no shutdown**) before a **no shutdown** on a sub-interface is executed.

Syntax

no shutdown

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATM*xx.x*>)#**

Example

The following example sets the ATM sub-interface to the administrative up state:

```
XSR(config-if<ATM0/1/0>)#no shutdown
```

oam-pvc

This command enables end-to-end F5 (circuit) OAM cell procedures for ATM Permanent Virtual Circuit (PVC) management. OAM cells and how they are used are as follows:

- *Alarm Indication Signal* (AIS) – Received from the network to indicate a problem in the forward-to-XSR data flow.
- *Continuity Check* (CC) – Echoed to the sender when received. The XSR does not generate CC cells for connectivity management but will respond to CC procedure negotiation cells.
- *Loopback* – Echoed back to the sender when received. The XSR sends loopback cells to monitor the end-to-end connectivity on the VC.
- *Remote Defect Indication* (RDI) – Received from the network to indicate a problem in the reverse-from-XSR data flow. Sent to the network to indicate a problem in the local node XSR as well as in response to any AIS cells received.

The loopback cells monitor and declare the circuit up or down as follows:

- The circuit is UP immediately after line training completes successfully.
- The circuit is declared DOWN when down-count consecutive loopback response cells are missed.
- The circuit is declared UP when up-count consecutive loopback response cells are received.

This command requires a properly configured ATM sub-interface.

Syntax

oam-pvc [*manage*][*frequency*]

manage	Optional keyword.
frequency	Interval between sending end-to-end F5 OAM loopback cells when the VC is in the UP state. Range: 1 to 3600 seconds.

Syntax of the "no" Form

The *no* form of this command disables all OAM procedures for this sub-interface:

no oam-pvc

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATM*x*/*x*/*x*.*x*>)#**

Defaults

- OAM procedures: Disabled
- Interval between loopback cells (frequency): 10 seconds
- Initial down-count value: 5
- Initial up-count value: 3

Example

The following example sets the OAM frequency to *20* seconds:

```
XSR(config-if<ATM0/1/0.1>)#oam-pvc manage 20
```

oam retry

This command configures parameters related to OAM cell handling for ATM VC management. This command requires a properly configured ATM sub-interface.

Syntax

```
oam retry up-count down-count retry-frequency
```

up-count	Sum of consecutive end-to-end F5 OAM loopback cells responses that must be received to change the VC connection state to up. Range: 0 to 255.
down-count	Sum of consecutive end-to-end F5 OAM loopback cells responses that are not received to change the VC connection state to down. Range: 0 to 255.
retry-frequency	Interval between sending end-to-end F5 OAM loopback cells when a change in the up/down state of a VC is being verified. Range: 1 to 3600 seconds.

Syntax of the "no" Form

The *no* form of this command returns this parameter to its default setting:

```
no oam retry
```

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATMx/x/x.x>)#**

Default

- Initial down-count value: 5
- Initial up-count value: 3
- Initial retry-frequency value: 10
- Default settings apply only when OAM management has been enabled with the **oam-pvc** command.

Example

This example sets the up-count to *5*, the down-count to 8, and the retry-frequency to *2* seconds:

```
XSR(config-if<ATM0/1/0.1>)#oam retry 5 8 2
```

pvc

This command sets the sub-interface circuit type to PVC and specifies ATM VPI/VCI values. This command requires a properly configured ATM sub-interface.

Syntax

pvc vpi/vci

vpi/vci	ATM VC identifier values. VPI range: 0 to 255, VCI range: 0 to 65535.

Syntax of the "no" Form

The *no* form of this command returns this parameter to its default setting:

no pvc

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATMx/x/x.x>)#**

Default

VPI/VCI defaults to 1/32. This is not the ILMI virtual circuit.

Example

This example sets the sub-interface circuit type to *PVC* and sets the ATM VPI/VCI values to *2/48*:

```
XSR(config-if<ATM0/1/0.1>)#pvc 2/48
```

shutdown

This command sets the ATM sub-interface to the administrative *Down* state halting all data traffic on this VC.

Syntax

shutdown

Syntax of the "no" Form

Refer to the **atm sub-interface** command on page 13-88.

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATMx/x/x.x>)#**

Example

The following example sets the ATM sub-interface to the administrative down state:

```
XSR(config-if<ATM0/1/0.1>)#shutdown
```

no shutdown

This command sets the ATM interface to the administrative *Up* state and enables the line for operation. Data traffic cannot flow until at least one associated sub-interface is set to the administrative Up state. Issuing this command does not change the administrative state of sub-interfaces associated with this ATM interface.

This command surveys the status of the DSP firmware (which was loaded and started at boot time) and if it finds it in an illegal state (i.e., crashed), it reloads and restarts the DSP firmware before proceeding with the **no shutdown** operation. Depending on the size of the DSP firmware and characteristics of the download process, this operation may take a noticeable length of time.

Syntax

no shutdown

Mode

ATM Interface configuration: **XSR(config-if<ATMxx>)#**

Example

The following example sets the ATM interface to the administrative *up* state:

```
XSR(config-if<ATM0/1/0>)#no shutdown
```

shutdown

This command sets the ATM interface to the administrative *Down* state. As a result, all ATM sub-interfaces associated with this ATM interface are shut down, all data traffic is stopped and the line disabled.

Syntax

shutdown

Syntax of the “no” Form

Refer to **no shutdown** on page 13-98.

Mode

ATM Interface configuration: **XSR(config-if<ATMxx>)#**

Example

The following example sets the ATM interface to the administrative down state:

```
XSR(config-if<ATM0/1/0>)#shutdown
```

PPP Configuration Commands

This section lists the subset of PPP configuration commands that apply when an ATM sub-interface is configured for PPPoA or PPPoE encapsulation.

ppp chap

This command configures PPP to use the Challenge Handshake Authentication Protocol (CHAP) for user authentication on a PPP session. This command requires a properly configured ATM sub-interface specifying encapsulation type PPPoA or PPPoE.

Syntax

ppp chap {**hostname** <*name*> | **password** *pwd* | **refuse**}

name	Specifies the CHAP hostname.
pwd	Specifies the CHAP password as *pwd*.
refuse	Rejects authentication by CHAP.

Syntax of the “no” Form

The *no* form of this command returns this parameter to its default setting:

no ppp chap

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATM*x*/*x*/*x*.*x*>)#**

Default

Disabled

Example

The following example designates the CHAP hostname ENT1:

XSR(config-if<ATM0/1/0.1>)#ppp chap hostname ENT1

ppp keepalive

This command enables PPP to use LCP echo requests as a keepalive mechanism. It requires a properly configured ATM sub-interface specifying encapsulation type PPPoA or PPPoE.

Syntax

ppp keepalive <*seconds*>

seconds	Interval between keepalive messages, ranging from 0 to 32767 seconds.

Syntax of the "no" Form

The *no* form of this command returns this parameter to its default setting:

no ppp keepalive

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATMx/*x*/*x*.*x*>)#**

Defaults

- Disabled
- Keepalive period: 30 seconds

Example

This example enables the keepalive mechanism and sets the time between messages to 20 seconds:

XSR(config-if<ATM0/1/0.1>)#ppp keepalive 20

ppp lcp

This command configures Link Control Protocol (LCP) parameters for PPP. It requires a properly configured ATM sub-interface specifying encapsulation type PPPoA or PPPoE.

Syntax

ppp lcp {**max-configure** <*count1*> | **max-failure** <*count2*> | **max-terminate** <*count3*>}

max-configure *count1*	Peak number of Configure-Requests to send. Range: 1 to 255.
max-failure *count2*	Peak number of Configure-Nak packets to send. Range: 1 to 255.
max-terminate *count3*	Peak number of Terminate-Requests to send. Range: 1 to 255.

Syntax of the "no" Form

The *no* form of this command returns this parameter to its default setting:

no ppp lcp

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATMx/*x*/*x*.*x*>)#**

Defaults

- *Configure-Requests:* 10
- *Configure-Nak*: 5
- *Terminate-Requests*: 2

Example

The following example sets LCP parameters:

```
XSR(config-if<ATM0/1/0.1>)#ppp lcp max-configure 5 max-failure 5 max-terminate 2
XSR(config-if<ATM0/1/0.1>)#
```

ppp max-bad-auth

This command configures the maximum number of authentication failures for PPP. It requires a properly configured ATM sub-interface specifying encapsulation type PPPoA or PPPoE.

Syntax

ppp max-bad-auth <*count*>

count	Peak number of authentication attempts. Range: 0 to 4,294,967,295

Syntax of the "no" Form

The *no* form of this command returns this parameter to its default setting:

no ppp max-bad-auth

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATMx/x/x.x>)#**

Default

Default number of attempts: 0

Example

The following example resets the command parameter to 16:

```
XSR(config-if<ATM0/1/0.1>)#ppp max-bad-auth 16
```

ppp pap

This command configures PPP to use the Password Authentication Protocol (PAP) for user authentication on a PPP session. This command requires a properly configured ATM sub-interface specifying encapsulation type PPPoA or PPPoE.

Syntax

ppp pap sent-username <*username*> **password** <*userpassword*>

username	The name to use for authentication.
userpassword	The user's password.

Syntax of the “no” Form

The *no* form of this command returns this parameter to its default setting:

no ppp pap

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATMx/*x*/*x*.*x*>)#**

Default

PAP is disabled

Example

The following example sets the PAP user name to *bob* and the password to *confidential*:

XSR(config-if<ATM0/1/0.1>)#ppp sent-name bob password confidential

ppp quality

This command configures the minimum link quality for PPP, which is a measure of the amount of data successfully passed over the link. The minimum quality value is specified as a percentage of the total data sent. This command requires a properly configured ATM sub-interface specifying encapsulation type PPPoA or PPPoE.

Syntax

ppp quality *<percent>*

percent	The minimum link quality value, ranging from 0 to 100.

Syntax of the “no” Form

The *no* form of this command returns this parameter to its default setting:

no ppp quality

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATMx/*x*/*x*.*x*>)#**

Default

Disabled

Example

The following example sets the minimum link quality value to *88*%:

XSR(config-if<ATM0/1/0.1>)#ppp quality 88

ppp timeout retry

This command sets the maximum time to wait for a response during PPP negotiation. It requires a properly configured ATM sub-interface specifying encapsulation type PPPoA or PPPoE.

Syntax

ppp timeout retry *<seconds>*

seconds	The peak wait interval, ranging from 1 to 255 seconds.

Syntax of the "no" Form

The *no* form of this command returns this parameter to its default setting:

no ppp timeout retry

Mode

ATM Sub-Interface configuration: **XSR(config-if<ATMx/x/x.x>)#**

Default

3 seconds

Example

This example resets the maximum wait time for a response during PPP negotiation to *12* seconds:

```
XSR(config-if<ATM0/1/0.1>)#ppp timeout retry 12
```

ATM Clear and Show Commands

clear counters atm

This command clears ATM counters for the ATM interface.

Syntax

clear counters atm {*slot/card/port*}

slot	The XSR slot number, ranging from 0 to 2.
card	The XSR NIM number, ranging from 1 to 2.
port	The XSR slot number: 0. The sub-interface number ranges from 1 to 30.

Mode

Privileged EXEC: **XSR#**

Example

The following example clears the ATM counters:

```
XSR#clear counters atm
```

show controllers atm

This command displays internal hardware configuration and operational interface details regarding: receive (Rx) and transmit (Tx) DMA descriptors, memory usage, and PCI device ID information. When you issue the command to display *sub-interface* statistics, the output returned includes: packet processor (QOS) scheduling statistics, ATM sub-interface counters, ATM sub-interface data plane status, and driver circuit statistics.

Syntax

show controllers atm {*slot/card/port.sub-interface*}

slot	The XSR slot number, ranging from 0 to 2.
card	The XSR NIM number, ranging from 1 to 2.
port	The XSR slot number: 0.
sub-interface	Identifies a sub-interface on that interface, ranging from 1 to 30.

Mode

EXEC or Privileged EXEC: **XSR>** or **XSR#**

Examples

The following is sample output when an *interface* is specified:

```
XSR#show controllers atm 1/0

********** ATM Controller Stats **********
ATM 1/0

DSP Image File: CFlash:adsl.fls
DSP File Rev. : 1.0.0.1
DSP Image Rev.: 43e2ea93

Attenuation: 43.0 db        SNR Margin: 6 db            CRC Errors: 0
DMT state: 42

OAM counters:               UNK counters:               Cells:
ifInOctets      00000000    ifInOctets      00000000    AIS in      00000000
ifInUcastPkts   00000000    ifInUcastPkts   00000000    RDI in      00000000
ifInDiscards    00000000    ifInDiscards    00000000    RDI out     00000000
ifInErrors      00000000    ifInErrors      00000000    CC in       00000000
ifOutOctets     00000000    ifOutOctets     00000000    CC out      00000000
ifOutUcastPkts  00000000    ifOutUcastPkts  00000000    LBBK in     00000000
ifOutDiscards   00000000    ifOutDiscards   00000000    LPBK out    00000000
ifOutErrors     00000000    ifOutErrors     00000000
total_count     0
tx_notready     0
tx_toomany      0
```

The following is sample output when a *sub-interface* is specified:

```
XSR#show controllers atm 1/0.1

********** ATM Sub-Interface Stats **********
ATM 1/0.1

Packet Processor Tx Scheduler Stats:
        952 Packet driver Tx OK
        0 Packet driver not Tx: MUX END_ERR_BLOCK
        0 Packet driver not Tx: MUX ERROR
        0 Packet driver not Tx: Unknown Msg from MUX

Statistic Counters:
        Rx PacketTotalCount     987
        Rx PacketDiscardCount   18
        Rx MuxHeaderError       0
        Rx SnapHeaderError      0
        Rx PPPoEethTypeError    0
        Rx PPPoEethTypeARP      6
        Rx PPPoEethTypeIP       12
        Rx PPPoEethTypeRARP     0
        Tx PacketTotalCount     952
        Tx PacketDiscardCount   0

********** ATM Data Object Stats **********
Upper Adjacent is CONNECTED and UP,  ATM PassData is TRUE
FE: Admin Up / Oper Up  PPPoE: Oper Up

********** Driver Virtual Circuit Stats **********
VPI/VCI 1/32:
ccRx1               987
ccRx2               987
received-adslr1     987
noeop               0
crc                 0
wor                 0
ovr                 0
toomany             0
stop                0
be1                 0
be2                 0
receivertnerr       0
nonewmblk           0
receivertnnull      0
tx_null_mblk        0
tx_no_enable        0
tx_length_err       0
sent-adslt          952
tx_no_free_slots    0
tx_no_showtime_loop 0
```

Parameters in the Sub-Interface Response

DSP Image File: CFlash:adsl.fls	Name of the file containing the DSP image.
DSP Image Rev.: 43e2ea93	Vendor's revision of the DSP image.
DMT state: 42	Current operational state of the DSP.
OAM counters/ UNK counters	Sub-set of the interface table input and output counters for the OAM and unconfigured channels on the ATM interface. Refer to RFC-1213 for parameter descriptions.
Cells:	Detailed OAM cell totals for receive and transmit counters.
total_count/ tx_notready/tx_toomany	Internal chipset debug counters.

Packet Processor Tx Scheduler Stats

952 Packet driver Tx OK	Sum of packets transmitted.
0 Packet driver not Tx: MUX END_ERR_BLOCK	Sum of failed transmit attempts due to the driver returning an END_ERR_BLOCK status.
0 Packet driver not Tx: MUX ERROR	Sum of failed transmit attempts due to the driver returning an ERROR status.
0 Packet driver not Tx: Unknown Msg from MUX	Sum of failed transmit attempts due to the driver returning an unknown error status.

ATM Sub-interface Statistic Counters:

Rx PacketTotalCount	Sum of packets received.
Rx PacketDiscardCount	Sum of packets received that were discarded because of an error.
Rx MuxHeaderError	Sum of packets received that were discarded due to an error in the VC Multiplexing encapsulation header.
Rx SnapHeaderError	Sum of packets received that were discarded due to an error in the LLC/SNAP encapsulation header.
Rx PPPoEethTypeError	Sum of PPPoE packets received that were discarded because the Ethernet type is unsupported.
Rx PPPoEethTypeARP	Sum of PPPoE packets received that were discarded because the Ethernet type ARP is unsupported.
Rx PPPoEethTypeIP	Sum of PPPoE packets received that were discarded because the Ethernet type IP is unsupported.
Rx PPPoEethTypeRARP	Sum of PPPoE packets received that were discarded because the Ethernet type RARP is unsupported.
Tx PacketTotalCount	Sum of packets transmitted.
Tx PacketDiscardCount	Sum of transmit packets discarded for any reason.
ATM Data Object Stats	Internal data plane status information.
VPI/VCI 1/32	Virtual Path Index and Virtual Circuit Index for the ATM PVC.
ccRx2 987 through *tx_no_showtime_loop 0*	Driver internal debug counters.

show interface atm

This command displays the running configuration and statistical details for an ATM interface. Statistics supported by the ADSL interface are hardware dependent. General categories include the following:

- Analog details including upstream and downstream bit rates
- ATM cell counters (especially OAM cells)
- OAM (circuit UP/DOWN) state
- Frame (AAL5) counters
- Layer state information
- VC table
- Administrative state (Enabled/Disabled)
- Operational state (Up/Down)
- Loopback on
- DSP firmware
- Backup interface
- Description string

When you issue the command to display sub-interface statistics, the output returned includes:

- VPI/VCI
- IP address (value + configured or negotiated)
- Encapsulation method
- Administrative state (enabled/disabled)
- Operational state (Up/Down)
- PPP state information (PPPoE - host name/service name)
- Description string
- VPN information

Syntax

show interface atm {*slot/card/port.sub-interface*}

slot	The XSR slot number, ranging from 0 to 2.
card	The XSR NIM number, ranging from 1 to 2.
port	The XSR slot number: 0.
sub-interface	Identifies a sub-interface on that interface, ranging from1 to 30.

Mode

EXEC or Privileged EXEC: **XSR>** or **XSR#**

Examples

The following is sample output when an *interface* is specified:

```
XSR#show interface atm 1/0

********** ATM Interface Stats **********
ATM 1/0 is Admin Up / Oper Up

The name of this device is adsl

Administrative State is ENABLED

Operational State is UP
OAM circuit is UP

The upstream data rate is 480 kbit/sec
The downstream data rate is 10208 kbit/sec


General info:
ifindex              0
ifType               0
ifAdminStatus        1
ifOperStatus         1
ifLastChange         00:02:34
ifInOctets           2950
ifInUcastPkts        47
ifInNUcastPkts       0
ifInDiscards         0
ifInErrors           0
ifInUnknownProtos    0
ifOutOctets          5088
ifOutUcastPkts       48
ifOutNUcastPkts      0
ifOutDiscards        0
ifOutErrors          0
ifOutQLen            100
AAL5 in              47
AAL5 out             48
HEC errors           0
AIS  F4              0
RDI  F4              0
CC   F4              0
LPBK F4              0

VPI/VCI      AAL5        AIS         RDI          CC        LPBK     AIS/RDI
1/32     00000047   00000000   00000000   00000000   00000000
```

The following is sample output when a *sub-interface* is specified:

```
XSR#show interface atm 1/0.1

********** ATM Sub-Interface Stats **********
ATM 1/0.1 is Admin Up / Oper Up

Internet address is 30.0.0.11, subnet mask is 255.255.255.255
LCP       State: OPENED
IPCP      State: OPENED

PPPoE is Oper Up
```

```
The logical link is currently Up
The Name of the Access Concentrator is ENTERASY-CDDU1S
The Session Id is 0x000b
The MAC Address of the Access Concentrator is  0x00:60:f9:11:01:08
The MTU is 1492

The name of this device is adsl-0

Administrative state is ENABLED

Operational State is UP
Circuit monitoring enabled

VPI is 1.
VCI is 32.

        ifindex             0
        ifType              0
        ifAdminStatus       1
        ifOperStatus        1
        ifLastChange        00:02:34
        ifInOctets          20510
        ifInUcastPkts       408
        ifInNUcastPkts      0
        ifInDiscards        0
        ifInErrors          0
        ifInUnknownProtos   0
        ifOutOctets         37728
        ifOutUcastPkts      388
        ifOutNUcastPkts     0
        ifOutDiscards       0
        ifOutErrors         0
        ifOutQLen           100
```

Parameters in the Interface Response

ATM 1/0 is Admin Up / Oper Up	Administrative state: Admin Up or Admin Down and Operational state: Oper Up or Oper Down.
The name of this device is adsl-0	Hardware device name.
Administrative State is ENABLED	Driver administrative state: ENABLED or DISABLED.
Operational State is UP	Driver operational state is UP or DOWN.
OAM circuit is UP	Driver OAM channel state is UP or DOWN.
The upstream data rate is 480 kbit/ sec.	Negotiated upstream data rate.
The downstream data rate is 10208 kbit/sec.	Negotiated downstream data rate.

General info:	MIB2 interface table entries as described in RFC-1213 including AIS F4, RDI F4, CC F4, LPBK F4. The last four fields in the General info section count the number OAM cells (by type) received by the interface on the Virtual Path (F4) flow. The circuit table at the end of the display lists all the configured ATM sub-interfaces related to this ATM interface. • VPI/VCI - PVC circuit identifier. • AAL5 - Sum of AAL5 frames received. • AIS - Sum of received Alarm Indication Signal cells received. • RDI - Sum of Remote Defect Indication cells received. • CC - Sum of Continuity Check cells received. • LPBK - Sum of Loopback cells received. • AIS/RDI - the current alarm state of the circuit: AIS or RDI

Parameters in the Sub-Interface Response

ATM 1/0.1 is Admin Up / Oper Up	Administrative state: Admin Up or Admin Down; Operational state: Oper Up or Oper Down.
Internet address is 30.0.0.11, subnet mask is 255.255.255.255	IP layer information.
LCP State: OPENED/IPCP State: OPENED	PPP layer information.
PPP Layer Information	
PPPoE is Oper Up	
The logical link is currently Up	
The Name of the Access Concentrator is ENTERASY-CDDU1S	
The Session Id is 0x000b	
The MAC Address of the Access Concentrator is 0x00:60:f9:11:01:08	
The MTU is 1492	
The name of this device is adsl-0	Hardware device name.
Administrative state is ENABLED	Driver administrative state: ENABLED or DISABLED.
Operational State is UP	Driver operational state: UP or DOWN
Circuit monitoring enabled/Circuit monitoring disabled	Circuit monitoring operational state. This line will only be displayed when OAM procedures are enabled by the **OAM-PVC** command and the ADSL line is UP .
VPI is 1/VCI is 32	Virtual Path Index and Virtual Circuit Index for the ATM PVC.

The last section contains the MIB2 interface table as described in RFC-1213.

14

Configuring the VPN

Observing Syntax and Conventions

The CLI Syntax and conventions use the notation described in the following table.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[x]	[ ] Square brackets indicate an optional parameter (*italic*)
[x | y | z]	[|] Square brackets with vertical bar indicate a choice of values
{x | y | z}	{ | } Braces with vertical bar indicate a choice of a required value
[x {y | z}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<*xx*>)	*xx* signifies the interface type and number; e.g., **F1**, **G3**, **S2/1.0**, **D1. F** indicates a FastEthernet, and **G** a GigabitEthernet port.
XSR(aaa-method-*xx*)#	*xx* signifies the AAA Method type; e.g., **local**, **pki**, **radius**
Next Mode entries display the CLI prompt after a command is entered.	
Sub-command headings are displayed in ***red***, italicized text.	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

VPN Commands

The following command subsets configure the Virtual Private Network suite of functionality for the XSR:

- "PKI commands" on page 14-84.
- "CA Identity Mode Commands" on page 14-84.
- "Other Certificate Commands" on page 14-90.
- "IKE Security Protocol Commands" on page 14-94.
- "ISAKMP Protocol Policy Mode Commands" on page 14-95.
- "Remote Peer ISAKMP Protocol Policy Mode Commands" on page 14-99
- "Remote Peer Show Commands" on page 14-104.
- "IPSec Commands" on page 14-106.
- "IPSec Clear and Show Commands" on page 14-108.

- "Crypto Map Mode Commands" on page 14-110.
- "Crypto Transform Mode Commands" on page 14-115.
- "Crypto Show Commands" on page 14-118.
- "Interface CLI Commands" on page 14-121.
- "Interface VPN Commands" on page 14-122.
- "Tunnel Commands" on page 14-127.
- "Tunnel Clear and Show Commands" on page 14-132.
- "Additional Tunnel Termination Commands" on page 14-134.
- "DF Bit Commands" on page 14-137.

Note: AAA commands are described in Chapter 13: *Configuring Security*.

PKI commands

The following commands configure Public Key Infrastructure (PKI) on the XSR.

CA Identity Mode Commands

crypto ca identity

This command declares the Certificate Authority (CA) the XSR should use and identifies CAs which may be required as part of the CA chain for the router or a peer IPSec client. If you previously declared the CA and just want to update its characteristics, specify the name you previously created. In some cases, the CA might require a particular CA name, such as its domain name.

Performing this command acquires CA Identity mode, where you can specify CA characteristics with the following sub-commands:

- **`crl frequency`** - Specifies the interval between Certificate Revocation List (CRL) retrievals and other maintenance that may be performed periodically. Refer to page 14-85 for the command definition.
- **`enrollment http-proxy`** - Specifies the local HTTP proxy server. It is optional. Refer to page 14-86 for the command definition.
- **`enrollment retry count`** - Specifies how many certificate enrollment polls the XSR will send before giving up. It is defaulted. Refer to page 14-86 for the command definition.
- **`enrollment retry period`** - Specifies an interval that the XSR should wait between sending certificate request retries. It is defaulted. Refer to page 14-87 for the command definition.
- **`enrollment url`** - Specifies the URL of the CA and is always required. Refer to page 14-88 for the command definition.

Syntax

`crypto ca identity` *`name`*

name	Name for the CA.

Syntax of the "no" Form

Use the *no* form to delete all identity information and certificates associated with the CA:

no crypto ca identity *name*

Mode

Global configuration: **XSR(config)#**

Next Mode

Certificate Authority Identity configuration: **XSR(ca-identity)#**

Examples

The following example declares and identifies characteristics of the CA. In this example, the name *ACMEca* is created for the CA, which is located at **http://ca_server**.. This is the minimum configuration required to declare a CA.

```
XSR(config)#crypto ca identity ACMEca
XSR(ca-identity)#enrollment url http://ca_server
```

The following example sets a nonstandard retry period and count, and permits the router to accept certificates when CRLs are not obtainable.

```
XSR(config)#crypto ca identity ACMEca
XSR(ca-identity)#enrollment url http://AAA_ca/coldstorage/scripts.exe
XSR(ca-identity)#query url ldap://serverx
XSR(ca-identity)#enrollment retry period 20
XSR(ca-identity)#enrollment retry count 100
```

In the example above, if the XSR does not get a certificate back from the CA within *20* minutes of sending a certificate request, it will resend the request. The XSR will repeat certificate requests every retry period until until *100* requests have been sent. If the CA is not available at the specified location, obtain the URL from your CA administrator.

crl frequency

The command specifies the interval between Certificate Revocation List (CRL) retrievals.

Syntax

crl frequency *number*

numbers	Interval between retries, ranging from 1 to 1440 minutes.

Syntax of the "no" Form

The *no* form of this command resets the value to the default:

no crl frequency

Mode

Certificate Authority Identity configuration: **XSR(ca-identity)#**

Example

The following example sets the CRL to be retrieved for *five* hours:

```
XSR(config)#crypto ca identity ACMEca
XSR(ca-identify)crl frequency 300
```

enrollment http-proxy

This command specifies the local HTTP proxy server name and port.

Syntax

enrollment http-proxy *hostname port_#*

hostname	The URL of the local HTTP proxy server, which is the proxy server's IP address.
port_#	HTTP Proxy server port number, ranging from 1 to 10,000.

Syntax of the "no" Form

The *no* form of this command clears the proxy server setting:

```
no enrollment http-proxy
```

Mode

Certificate Authority Identity configuration: **XSR(ca-identity)#**

Example

The following example sets the HTTP proxy server IP address and port #:

```
XSR(config)#crypto ca identity ACMEca
XSR(ca-identity)#enrollment http-proxy 192.168.57.9 999
```

enrollment retry count

This command specifies how many times the XSR resends a certificate request when it does not receive a certificate from the Certificate Authority (CA) from the previous request.

Syntax

enrollment retry count *number*

number	Attempts the XSR will make to resend a certificate request to the CA while waiting on an original request. Range: 1 to 100.

Syntax of the "no" Form

The *no* form of this command resets the value to the default:

```
no enrollment retry count
```

Default

3

Mode

Certificate Authority Identity configuration: **XSR(ca-identity)#**

Example

The following example declares a CA, and changes the retry period to *10* minutes and the retry count to *60*. The XSR will resend the certificate request every 10 minutes until it receives the certificate or until approximately 10 hours pass since the original request was sent, whichever occurs first. (10 minutes x 60 tries = 600 minutes [10 hours]).

```
XSR(config)#crypto ca identity ACMEca
XSR(ca-identity)#enrollment url http://ca_server
XSR(ca-identity)#enrollment retry period 10
XSR(ca-identity)#enrollment retry count 60
```

enrollment retry period

This command specifies the wait period between certificate requests.

Syntax

enrollment retry period *minutes*

minutes	The interval, ranging from 1 to 60 minutes, the XSR waits before resending a certificate request to the CA.

Syntax of the "no" Form

Use the *no* form of the command to reset the retry period to the default:

```
no enrollment retry period
```

Default

5 minutes

Mode

Certificate Authority Identity configuration: **XSR(ca-identity)#**

Example

The following example declares a CA and changes the retry period:

```
XSR(config)#crypto ca identity ACMEca
XSR(ca-identity)#enrollment url http://ca_server
XSR(ca-identity)#enrollment retry period 5
```

enrollment url

This command sets the Uniform Resource Locator (URL) of the Certificate Authority (CA). If the CA *cgi-bin* script site is not the default */cgi-bin/ pkiclient.exe* at the CA, you must also include the non-standard script site in the URL as *http://CA_name/ script_location* where *script_location* is the full path to the CA scripts. Be aware that the URL format may vary.

Syntax

enrollment url *url*

url	The URL of the CA where the XSR sends certificate requests. The URL may be in the form of *http://CA_name* where *CA_name* is the CA's host IP address or defined static IP hostname.

Syntax of the "no" Form

This command's *no* form deletes the CA's URL value from the configuration:

no enrollment url *url*

Mode

Certificate Authority Identity configuration: **XSR(ca-identity)#**

Examples

The following example shows the minimum configuration required to declare a CA:

```
XSR(config)#crypto ca identity ACMEca
XSR(ca-identity)#enrollment url http://ca_server
```

The example below shows a static IP hostname for the enrollment URL:

```
XSR(config)#crypto ca identity CAserver
XSR(ca-identity)#enrollment url http://ParentCA.domain.com/ certsrv/mscep/
mscep.dll
```

crypto ca enroll

This command enrolls a certificate for the XSR with the specified Certificate Authority (CA). It is not saved in the XSR configuration file but in a local encrypted database named **cert.dat.**

Notes: You can remove existing certificates with the **no certificate** command.

If an enroll request to the Entrust CA fails, be sure the CA does not contain an outstanding PENDING enroll request from that same XSR by a previously incomplete enroll request. Because the Entrust CA allows only one outstanding request from any single client seeking certificate enrollment, the CA administrator must delete the pending certificate for the outstanding request at the CA then the XSR can reissue its certificate enrollment request.

For Verisign CA compliance, you must provide the *domain name* that you specified when signing up with Verisign by using the **ip domain** command. See page 5-155 for command details.

Caution: We recommend that you do not enroll more certificates than permitted by the 1.5 MByte system limit imposed on the **cert.dat** Flash file. Doing so may destabilize the XSR and require you to delete the file.

Syntax

crypto ca enroll *name*

name	Name of the CA. Use the same name as when you declared the CA with the **crypto ca identity** command.

Syntax of the "no" Form

The *no* form of this command cancels a current enrollment request:

no crypto ca enroll *name*

Mode

Global configuration: **XSR(config)#**

Sample Output

The following script displays when you invoke the **crypto ca enroll** command. Note that you are prompted to enter your password and whether to proceed.

```
XSR(config)#crypto ca enroll ACMEca
%
% Start certificate enrollment
% Create a challenge password. You will need to verbally provide this password to
the CA Administrator in order to revoke your certificate.
For security reasons your password will not be saved in the configuration.
Please make a note of it.
Password:****
Re-enter password:****

Include the router serial number in the subject name (y/n) ? y
The serial number in the certificate will be: 3526015000250142
Request certificate from CA (y/n) ? y
You may experience a short delay while RSA keys are generated.
Once key generation is complete, the certificate request
will be sent to the Certificate Authority.
Use 'show crypto ca certificate' to show the fingerprint.
<186>Aug 29 7:11:1 192.168.1.33 PKI: A certificate was successfully
received from the CA.
```

show crypto ca identity

This command displays data about enrolled Certificate Authorities (CA).

Syntax

show crypto ca identity

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following output displays when you invoke the command:

```
XSR#show crypto ca identity

CA Identity - childca2
 Enrollment Information:
 Retry Period:          5 minutes
 Retry Count:           3
 Crl Frequency:         60 minutes

CA Identity - childca1
 Enrollment Information:
 Retry Period:          5 minutes
 Retry Count:           3
 Crl Frequency:         60 minutes
CA Identity - ldapca
 Enrollment Information:
 URL:            http://1.1.1.10/certsrv/mscep/mscep.dll/
 Retry Period:          5 minutes
 Retry Count:           3
 Crl Frequency:         60 minutes
```

Other Certificate Commands

crypto ca authenticate

This command authenticates the Certificate Authority (CA) by obtaining the CA's certificate. It acquires the CA certificate, computes the CA's fingerprint, and stores the certificate and fingerprint locally.

Syntax

crypto ca authenticate *name*

name	The name of the CA. This is the same name used when the CA was declared with the **crypto ca identity** command.

Mode

Global configuration: **XSR(config)#**

Sample Output

The following script prompts you to accept the certificate.

```
XSR#crypto ca authenticate ACMEca

Certificate has the following attributes:
Fingerprint: 0123 4567 89AB CDEF 0123
Do you accept this certificate? [yes/no] y
```

crypto ca certificate chain

This command invokes Certificate Chain mode. In this mode, you can delete a certificate by entering the **no certificate** commands. If you issue this command, you should also:

- Ask the CA administrator to revoke XSR's certificates at the CA; you must supply the challenge password you created when you first got the certificates with **crypto ca enroll**.
- Remove the XSR's certificates from the configuration using the **certificate** command.

Syntax

crypto ca certificate chain *name*

name	CA name. Use the same name you declared using **crypto ca identity**.

Mode

Global configuration: **XSR(config)#**

Next Mode

Certificate chain configuration: **XSR(config-cert-chain)#**

Example

This command acquires Certificate Chain mode in which a certificate can be added or removed. Note that the script prompts you to remove the certificate:

```
XSR(config)#crypto ca certificate chain ACMEca
XSR(config-cert-chain)#no certificate 0123456789ABCDEF0123456789ABCDEF
% Are you sure you want to remove the certificate [yes/no]? yes
% Be sure to ask the CA administrator to revoke this certificate.
```

crypto ca crl request

This command downloads a new Certificate Revocation List (CRL) from the specified Certificate Authority (CA), updating the CRL.

Syntax

crypto ca crl request *name*

name	CA name. Use the same name you declared using **crypto ca identity**.

Mode

Global configuration: **XSR(config)#**

Example

The following below immediately downloads the latest CRL to the router:

```
XSR(config)#crypto ca crl request
```

show crypto ca crls

This command displays data about Certificate Revocation Lists (CRL) issued by a Certificate Authority (CA).

Syntax

show crypto ca crls

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following output displays when you invoke the command:

```
XSR#show crypto ca crls

CRL -
 State:              VALID
 Version:            V2
 Issuer:             C=US, O=Enterasys, OU=VPN2, CN=Child CA2
 Valid From:         2002 Aug 20th, 18:45:21 GMT
 Valid To:           2002 Aug 20th, 20:20:21 GMT
 Issuing CDP:        http://childca2/CertEnroll/Child%20CA2.crl
 Crl Size:           512 bytes

CRL - issued by ldapca
 State:              VALID
 Version:            V2
```

```
 Issuer:            C=US, O=sml, CN=ldapca
 Valid From:        2002 Aug 20th, 18:26:01 GMT
 Valid To:          2002 Aug 20th, 20:01:01 GMT
 Issuing CDP:        ldap://ldapca.sml.com/CN=ldapca(6),CN=ldapca,CN=CDP,CN=Publ
ic%20Key%20Services,CN=Services,CN=Configuration,DC=sml,DC=com?certificateRevoc
ationList?base?objectclass=cRLDistributionPoint
 Crl Size:          365 bytes
```

show crypto ca certificates

This command lists information about the following:

- XSR certificate, if you have requested them from CAs (see the **crypto ca enroll** command).
- CA certificates, if you received them (refer to the **crypto ca authenticate** command).

Syntax

```
show crypto ca certificates
```

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Example

The following sample output shows two XSRs' certificates and the CA's certificate. In this example, special usage RSA key pairs were previously generated, and a certificate was requested and received for each key pair.

```
XSR>show crypto ca certificates

Certificate
Subject Name
Name: XSR.example.com
IP Address: 10.0.0.1
Status: Available
Certificate Serial Number: 428125BDA34196003F6C78316CD8FA95
Key Usage: Signature

Certificate
Subject Name
Name: XSR.example.com
IP Address: 10.0.0.1
Status: Available
Certificate Serial Number: AB352356AFCD0395E333CCFD7CD33897
Key Usage: Encryption

CA Certificate
Status: Available
Certificate Serial Number: 3051DF7123BEE31B8341DFE4B3A338E5F
Key Usage: Not Set
```

The following is sample output from the command when the CA supports an RA. In this example, CA and RA certificates were requested earlier by the **crypto ca authenticate** command.

```
XSR>show crypto ca certificates

CA Certificate
Status: Available
Certificate Serial Number: 3051DF7123BEE31B8341DFE4B3A338E5F
Key Usage: Not Set

RA Signature Certificate
Status: Available
Certificate Serial Number: 34BCF8A0
Key Usage: Signature

RA KeyEncipher Certificate
Status: Available
Certificate Serial Number: 34BCF89F
Key Usage: Encryption
```

IKE Security Protocol Commands

The following commands configure the Internet Key Exchange (IKE) Security Protocol on the XSR.

clear crypto isakmp

This command clears one or all active Internet Key Exchange connections.

Syntax

clear crypto isakmp [*connection-id*]

connection-id	Sets which connection to clear. If this argument is not used, all existing links will be cleared.

Mode

Privileged EXEC: **XSR#**

Example

The following output shows an IKE connection between two peers connected by interfaces *172.21.114.123* and *172.21.114.67*:

```
XSR#show crypto isakmp sa
Connection-ID    State      Source           Destination      Lifetime
1                QM_IDLE    172.21.114.67    172.21.114.123   2000
8                QM_IDLE    155.0.0.1        155.0.0.2        4000
```

The following example clears IKE connection *8*:

```
XSR#clear crypto isakmp 8
```

ISAKMP Protocol Policy Mode Commands

crypto isakmp proposal

This command defines an IKE proposal (policy) - a set of parameters used during IKE negotiation. It invokes ISAKMP protocol policy configuration mode where the following sub-commands are available to specify parameters in the proposal:

- **`authentication`** - Authentication method used by an IKE proposal. Refer to page 14-96 for the command definition.
- **`encryption`** - Encoding method used by an IKE proposal. Refer to page 14-97 for the command definition.
- **`group`** - Diffie-Hellman group type used by an IKE proposal. Refer to page 14-97 for the command definition.
- **`hash`** - Hash algorithm used by an IKE proposal. Refer to page 14-98 for the command definition.
- **`lifetime`** - SA interval used by an IKE proposal. Refer to page 14-99 for the command definition.

Many IKE proposals (policies) can be configured on each peer participating in IPSec. When IKE negotiation begins, it tries to find a common proposal (policy) on both peers; the common proposal contains exactly the same encryption, hash, authentication, and Diffie-Hellman values. The lifetime value does not necessarily have to be the same.

Syntax

`crypto isakmp proposal` *`name`*

`name`	Proposal name to be defined.

Syntax of the “no” Form

To delete an IKE proposal (policy), use the *no* form of this command:

`no crypto isakmp proposal` *`name`*

Defaults

The *DEFAULT* proposal contains these default values:

- Authentication: RSA signatures
- Encryption: Triple DES
- Group: 2
- Hash: SHA-1
- Lifetime: 28,840 seconds (8 hours)

Mode

Global configuration: **`XSR(config)#`**

Next Mode

ISAKMP protocol proposal configuration: **XSR(config-isakmp)#**

Example

The following example configures two policies for the peer:

```
XSR(config)#crypto isakmp proposal 57
XSR(config-isakmp)#hash md5
XSR(config-isakmp)#authentication rsa-sig
XSR(config-isakmp)#group2
XSR(config-isakmp)#lifetime 5000
XSR(config)#crypto isakmp policy 99
XSR(config-isakmp)#authentication pre-share
XSR(config-isakmp)#lifetime 10000
```

The above configuration results in the following policies:

```
XSR# show crypto isakmp proposal
```

Name	Authentication	Encrypt	Integrity	Group	Lifetime
57	RSASignature	DES	HMAC-MD5	Modp1024	5000
99	PreSharedKeys	DES	HMAC-SHA	Modp768	10000
DEFAULT	RSASignature	DES	HMAC-SHA	Modp768	86400

authentication

This command specifies the authentication method used within an IKE proposal (policy).

Syntax

authentication {*rsa-sig* | *pre-share*}

rsa-sig	RSA signatures public key authentication method.
pre-share	Pre-shared keys authentication method.

Syntax of the “no” Form

The *no* form of this command resets authentication to the default:

no authentication

Default

rsa-sig

Mode

ISAKMP protocol policy configuration: **XSR(config-isakmp)#**

Example

This example specifies RSA signatures authentication for IKE proposal *ACMEproposal*:

```
XSR(config)#crypto isakmp proposal ACMEproposal
XSR(config-isakmp)#authentication rsa-sig
```

encryption

This command sets the encryption algorithm used in an IKE proposal (policy).

Syntax

encryption {*des* | *3des* | *aes*}

des	Data Encryption Standard (DES) encryption.
3des	Triple Data Encryption Standard (3DES) encryption.
aes	Advanced Encryption Standard (AES) encryption.

Syntax of the "no" Form

The *no* form of this commands resets the algorithm to the default:

no encryption

Default

3DES

Mode

ISAKMP protocol proposal configuration: **XSR(config-isakmp)#**

Example

This example specifies *3DES* as the encryption method for the IKE proposal *ACMEproposal*:

```
XSR(config)#crypto isakmp proposal ACMEproposal
XSR(config-isakmp)#encryption 3des
```

group

This command sets the Diffie-Hellman group in an IKE proposal (policy).

Note: Due to the lack of an IETF standard, IKE Diffie-Helman bit groups 2048, 3072, and 4096 are not enabled.

Syntax

group {*1* | *2* | *5*}

1	768-bit Diffie-Hellman group.
2	1024-bit Diffie-Hellman group.
5	1536-bit Diffie-Hellman group.

Syntax of the "no" Form

The *no* form of this command resets the value to the default:

```
no group
```

Default

Group 2

Mode

ISAKMP protocol policy configuration: **XSR(config-isakmp)#**

Example

The following example configures Group 5 on *ACMEproposal*:

```
XSR(config)#crypto isakmp proposal ACMEproposal
XSR(config-isakmp)#Group5
```

hash

This command sets the hash algorithm used in an IKE proposal (policy).

Syntax

hash {*sha* | *md5*}

sha	Secure Hash Algorithm1 (SHA-1) hash.
md5	Message-Digest Algorithm (MD5) algorithm.

Syntax of the "no" Form

The *no* form this command resets to the default - *sha*:

```
no hash
```

Default

sha

Mode

ISAKMP Protocol Policy configuration: **XSR(config-isakmp)#**

Example

This example specifies *MD-5* as the hash algorithm to be used for IKE proposal *ACMEproposal*:

```
XSR(config)#crypto isakmp proposal ACMEproposal
XSR(config-isakmp)#hash md5
```

lifetime

This command specifies the lifetime of an IKE Security Association (SA) for a given IKE proposal (policy).

Syntax

lifetime *seconds*

seconds	The interval, in seconds, each SA exists before expiring.

Syntax of the "no" Form

The *no* form of this command resets to the default value:

no lifetime

Default

28,800 seconds (8 hours)

Mode

ISAKMP protocol policy configuration: **XSR(config-isakmp)#**

Example

The following example sets the IKE SA lifetime at *8 hours* for *ACMEproposal*:

```
XSR(config)#crypto isakmp proposal ACMEproposal
XSR(config-isakmp)#lifetime 28800
```

Remote Peer ISAKMP Protocol Policy Mode Commands

crypto isakmp peer

This command configures the remote peer's IP address and/or subnet and acquires ISAKMP configuration mode. The following sub-commands can be entered at ISAKMP Peer mode:

- **config-mode** sets the local IKE Mode configuration, the de facto standard to assign IP addresses within IKE. Refer to page 14-100 for the command definition.
- **exchange-mode** sets IKE to main or aggressive exchange mode. Refer to page 14-101 for the command definition.
- **nat-traversal** sets the IKE and IPSec NAT (Network Address Translation) traversal mode. Refer to page 14-102 for the command definition.
- **proposal** attaches IKE policies to a remote peer. Refer to page 14-102 for the command definition.
- **user-id** defines the identity information to be used during aggressive IKE Phase 1 negotiation. Refer to page 14-103 for the command definition.

Syntax

crypto isakmp *peer_address subnet-mask*

peer_address	Peer's IP address or IP subnet to which the policy will be attached.
subnet-mask	Value used with the *peer-address*.

Syntax

The *no* form of this command removes policies from a remote peer:

no crypto isakmp peer *peer_address subnet-mask*

Mode

Global configuration: **XSR(config)#**

Next Mode

Remote Peer ISAKMP protocol policy configuration: **XSR(config-isakmp-peer)#**

Example

The following example sets the remote peer's IKE policies:

```
XSR(config)#crypto isakmp peer 192.168.57.9 255.255.255.255
XSR(config-isakmp)#
```

config-mode

This command sets the local IKE Mode Configuration role. While not officially an IETF standard, *config-mode* is the de facto standard for assigning IP addresses within IKE.

Internet Key Exchange (IKE) Mode Configuration, as implemented by many vendors, allows a gateway to download an IP address (and other network level configuration) to the client as part of IKE negotiation. Using this exchange, the gateway gives IP addresses to the IKE client to be used as an *inner* IP address encapsulated under IPSec. This method provides a known IP address for the client that can be matched against IPSec policy.

When configured as a Mode Config *gateway*, the XSR allocates an IP address to a peerm requesting it and when configured as a *client*, the XSR requests an IP address from the gateway.

Syntax

config-mode {*client* | *gateway*}

client	Act as a Configuration Mode *client* with this peer.
gateway	Act as a Configuration Mode *server* with this peer.

Syntax of the "no" Form

The *no* form of this command resets IKE configuration mode to the default:

no config-mode

Default

Disabled

Mode

Remote Peer ISAKMP protocol policy configuration: **XSR(config-isakmp-peer)#**

Example

The following example configures the IKE IP address assignment mode to *client*:

```
XSR(config)#crypto isakmp peer 2.2.2.2 255.255.255.0
XSR(config-isakmp-peer)#config-mode client
```

exchange-mode

This command sets IKE to main or aggressive exchange mode.

Notes: It is useful to specify a *user ID* instead of an IP address when configuring an SA in aggressive mode (with pre-shared keys) for a peer whose IP address is dynamic. If you specify no ID, its IP address will be used by default. But, in that case, you will have to re-configure (with a new entry in the **aaa user** database) both ends of the tunnel every time the address changes. Use the **user-id <string>** command instead.

Due to the vulnerability of pre-shared keys on VPN devices using *aggressive* mode tunnels, Enterasys Networks recommends instead using a certificate or employing a very long password which is not listed in a dictionary.

Syntax

exchange-mode {*main* | *aggressive*}

main	IKE exchange mode set to main mode.
aggressive	IKE exchange mode set to aggressive mode.

Syntax of the “no” Form

The *no* form of this command resets the exchange mode to the default:

no exchange-mode

Default

Aggressive mode

Mode

Remote Peer ISAKMP protocol policy configuration: **XSR(config-isakmp-peer)#**

Example

The following example configures the IKE mode to *main*:

```
XSR(config)#crypto isakmp peer 192.168.57.9 255.255.255.255
```

```
XSR(config-isakmp-peer)#exchange-mode main
```

nat-traversal

The command sets the IKE and IPSec NAT (Network Address Translation) traversal mode used when communicating with remote peers matching the peer subnet and wildcard masks.

The *automatic* parameter configures IKE to automatically detect unroutable IP addresses between the local and remote gateway and to then switch to UDP encapsulation of IPSec traffic. The alternate values for this parameter (*enabled* and *disabled*) unconditionally turn UDP encapsulation of IPSec packets on or off, respectively.

Syntax

nat-traversal {**automatic** | **enabled** | **disabled**}

automatic	IKE NAT mode dynamically responds to discovered unroutable IP addresses by UDP-encapsulating this traffic.
enabled	IKE NAT mode *unconditionally on*.
disabled	IKE NAT mode *unconditionally off*.

Syntax of the “no” Form

The *no* form of this command resets the default value:

no nat-traversal

Default

Disabled

Mode

Remote Peer ISAKMP protocol policy configuration: **XSR(config-isakmp-peer)#**

Example

The following example sets IKE NAT mode to *enabled*:

```
XSR(config-isakmp-peer)#nat-traversal enabled
```

proposal

This command attaches up to three IKE policies to a remote peer. Proposals are configured with the **crypto isakmp proposal** command.

Syntax

proposal *pol1* [*pol12 pol13*]

pol2 pol13	Names of policies attached to the remote peer.

Syntax of the "no" Form

The *no* form of this command removes policies from the peer:

```
no proposal
```

Mode

Remote Peer ISAKMP protocol policy configuration: **`XSR(config-isakmp-peer)#`**

Example

The following example attaches a proposal to the remote peer:

```
XSR(config)#crypto isakmp peer 192.168.57.9 255.255.255.255
XSR(config-isakmp-peer)#proposal 3des_md5_gh2
```

user-id

This command defines the identity information to be used during *aggressive* IKE Phase 1 negotiation for peer-to-peer connections. Enter it when configuring the peer's ISAKMP for a peer with pre-shared keys whose IP address is *dynamic*. If you specify no ID, the *IP address* will be used by default. But, in that case, you will have to re-configure (with a new entry in the **`aaa user`** database) both ends of the tunnel every time the address changes.

Note: The exchange mode for this ISAKMP must be set to *aggressive*.

Syntax

```
user-id "string"
```

`"string"`	User-defined identification enclosed by quotations.

Syntax of the "no" Form

The *no* form of this command deletes the user identity:

```
no user-id "string"
```

Mode

Privileged EXEC: **`XSR#`**

Example

The following example configures the identification *ROBO1*. This ID will be used for aggressive IKE Phase 1 messages sent to the peer matching the ISAKMP's peer address (0.0.0.0, for example):

```
XSR(config)#crypto isakmp peer 0.0.0.0 0.0.0.0
XSR(config-isakmp-peer)#exchange-mode aggressive
XSR(config-isakmp-peer)#user-id "ROBO1 in Shrewsbury"
```

Remote Peer Show Commands

show crypto isakmp peer

This command displays attributes for each ISAKMP peer. IKE's first configuration derives from the IP address of the remote peer. ISAKMP peers created by EZ-IPSec configuration are marked with an asterisk (*) in the leftmost column of the **show** output. These proposals may not be used in other user-defined ISAKMP policies - they are reserved for EZ-IPSec.

Syntax

```
show crypto isakmp peer
```

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following is sample output from the command:

```
XSR#show crypto isakmp peer
Applicable Subnet   Exch-Mode  Config-Mode  NAT    User ID    Proposals
192.168.57.4/2      Main       Client       Off       p1      *** NONE ***
192.168.57.9/32     Main       Disabled     Off               *** NONE ***
```

The following output was produced by an ISAKMP peer created by EZ-IPSec:

```
XSR#show crypto isakmp peer
Applicable Subnet   Exch-Mode  Config-Mode  NAT    User ID    Proposals
* 141.154.196.87/32 Main       Client       Auto              ez-ike-3des-sha-rsa
                                                              ez-ike-3des-md5-rsa
```

Parameter Description

Applicable subnet	Subnet describing a range of IP addresses representing peers.
Applicable subnet	*Main* or *Aggressive*.
Config-Mode	*Client*, *Gateway* or *Disabled*.
NAT	Indicates whether NAT Traversal is *On* or *Off*. Be aware that Off may be indicated even when NAT-T is being used.
User ID	User-specified peer name.
Proposals	IKE policies.

show crypto isakmp proposal

This command lists attributes for each Internet Key Exchange (IKE) proposal. ISAKMP proposals created with EZ-IPSec are marked with an asterisk (*) in the **show** output. These proposals may not be used in other user-defined ISAKMP policies - they are reserved for EZ-IPSec.

Syntax

```
show crypto isakmp proposal
```

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

```
XSR#show crypto isakmp proposal
Name                    Authentication   Encrypt    Integrity Group      Lifetime
test                    PreSharedKeys    AES        HMAC-MD5  Modp1024
```

The following output was produced by ISAKMP proposals created via EZ-IPSec:

```
XSR#show crypto isakmp proposal
Name                    Authentication    Encrypt   Integrity   Group       Lifetime
*ez-ike-3des-sha-psk    PreSharedKeys     3DES      HMAC-SHA    Modp1024    28800
*ez-ike-3des-md5-psk    PreSharedKeys     3DES      HMAC-MD5    Modp1024    28800
*ez-ike-3des-sha-rsa    RSASignature      3DES      HMAC-SHA    Modp1024    28800
*ez-ike-3des-md5-rsa    RSASignature      3DES      HMAC-MD5    Modp1024    28800
```

show crypto isakmp sa

This command lists all current Internet Key Exchange Security Associations (SAs) for your XSR. An SA occupies a certain state depending upon where in the authentication process the peers are and what exchange mode they share - *Aggressive*, *Main* or *Quick*. During long exchanges, some of the MM states may be seen. Refer to the Parameter Descriptions for further explanation.

Syntax

```
show crypto isakmp sa
```

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following output displays two SAs, one in Main Mode exchange preparing to authenticate and the other in Quick Mode exchange ready for traffic:

```
XSR#show crypto isakmp sa
Connection-ID    State           Source           Destination      Lifetime
526              MM_KEY_AUTH     192.168.2.2      192.168.2.1
9                QM_IDLE         192.168.55.10    141.154.196.87
```

Parameters Descriptions

Main Mode Exchange

MM_NO_STATE	ISAKMP SA has only just been created and no state is yet established.
MM_SA_SETUP	Peers have agreed on settings for the ISAKMP SA.
MM_KEY_EXCH	Peers have exchanged Diffie-Hellman public keys and built a shared secret. The ISAKMP SA is *not* authenticated.
MM_KEY_AUTH	ISAKMP SA is *authenticated*. If the XSR began this exchange, this state transitions immediately to QM_IDLE and a Quick Mode exchange begins.

Aggressive Mode Exchange

AG_NO_STATE	ISAKMP SA has only just been created and no state is yet established.
AG_INIT_EXCH	Peers have made the first exchange in Aggressive Mode but the SA is not authenticated.
AG_AUTH	ISAKMP SA has been authenticated. If the XSR began this exchange, this state transitions immediately to QM_IDLE and a Quick Mode exchange begins.

Quick Mode Exchange

QM_IDLE	ISAKMP SA is quiescent. It remains authenticated with its peer and may be used for later Quick Mode exchanges.

IPSec Commands

This section describes commands that configure the IPSec protocol which provides anti-replay protection as well as data authentication and encryption.

access-list

This command creates an access list which is used to define which IP traffic will and will not be protected by the crypto process. ACLs associated with IPSec crypto map entries have these primary functions:

- Select outbound traffic to be protected by IPSec: the keyword permit equates with protected traffic.
- Indicate the data flow to be protected by the new Security Associations (SAs) - specified by a single permit entry- when initiating negotiations for IPSec SAs.
- Process inbound traffic to filter out and discard traffic that should have been protected by IPSec.
- Determine whether or not to accept requests for IPSec SAs on behalf of the requested data flows when processing IKE negotiation from the IPSec peer (negotiation is done only for *ipsec-isakmp crypto map* entries.) In order to be accepted, if the peer initiates IPSec negotiation, it must specify a data flow that is “permitted” by a crypto access list associated with an *ipsec-isakmp crypto map* entry.

Syntax

access-list *acl-number* {**deny** | **permit**} *protocol* [*source_addr source_mask* [**eq** *port*] *destination_addr destination_mask* [**eq** *port*]

acl-number	A uniquely defined access list number.
deny	Prevents traffic from being protected by IPSec in the contextof a particular crypto map entry: it does not allow the policy as set in **crypto map** statements to be applied to this traffic.
permit	Causes all IP traffic that matches the specified conditions to be protected by IPSec using the policy described by the corresponding **crypto map** command statements.
protocol	Name or number of an IP protocol. It can be one of the keywords *ip*, *tcp*, or *udp*, or an integer ranging from *1* to *254* representing an IP protocol number. To match any Internet protocol, including TCP, and UDP, use the keyword *ip*.
eq *port*	A clause to define a matching source and/or destination port number. Source and/or destination is defined by the location of the **eq** keyword in the command. A port number of zero matches any port. May only be used with TCP and UDP protocols.
source-addr	Address of the network or host from which the packet is sent.
source-mask	Netmask bits (mask) to be applied to *source_addr*.
destination-addr	IP address of the network or host to where the packet is sent.
destination-mask	Netmask bits (mask) to be applied to *destination_addr*.

Syntax of the “no” Form

The *no* form of this command removes the access list:

no access-list *acl-number* {**deny** | **permit**} *protocol* [*source_addr source_mask* [**eq** *port*] *destination_addr destination_mask* [**eq** *port*]

Default

An extended ACL defaults to a list that denies everything.

Mode

Global configuration: **XSR(config)#**

Examples

The following example configures two IP ACLs:

```
XSR(config)#access-list 100 permit ip 0.0.0.0 255.255.255.255 192.168.1.0
XSR(config)#access-list 101 permit ip 0.0.0.0 255.255.255.255 host 10.123.234.45
```

The following ACLs secure L2TP:

```
XSR(config)#access-list 120 permit udp any eq 1701 any
XSR(config)#access-list 130 permit udp any any eq 1701
```

IPSec Clear and Show Commands

clear crypto sa

This command deletes IPSec Security Associations (SAs) as follows:

- If the SAs were established via IKE, they are deleted and future IPSec traffic will require new SAs to be negotiated. (When IKE is used, the IPSec SAs are established only when needed.)
- The **peer** keyword deletes any IPSec SAs for the specified peer.
- The **map** keyword deletes any IPSec SAs for the named crypto map set.
- The **counters** keyword simply clears the traffic counters maintained for each SA; it does not clear the SAs themselves.

Note: If there are many thousands of tunnels in use, this command will use as many system resources as are available for as long as necessary to complete the task, making the XSR appear "frozen."

Syntax

```
clear crypto sa
clear crypto sa peer {ip-address | peer-name}
clear crypto sa map map-name
clear crypto sa counters
```

ip-address	Specify a remote peer's IP address.
peer-name	Specify a remote peer's name as the fully qualified domain name.
map-name	Specify the name of a crypto map set.

Default

If **peer**, **map**, or **counters** keywords are not used, all IPSec SAs are deleted.

Mode

Privileged EXEC: **XSR#**

Example

The following example clears the SA counters for *all* peers:

```
XSR#clear crypto sa counters
```

show access-lists

This command shows one or all access lists defined in the XSR. Alternatively, you can view the packet threshold after which the ACL violations log is triggered.

Syntax

```
show access-lists number log-update-threshold
```

number	Access list number defined using the **access-list** command.
log-update-threshold	Packet ceiling, when met, will trigger violations log.

Default

If an access list *number* is not specified, all access lists are shown.

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Examples

The following example displays configured access lists on the XSR:

```
XSR#show access-lists

Extended IP access list 100
        permit ip any host 192.168.1.0
```

The following example displays the log threshold:

```
XSR(config)#show access-lists log-update-threshold

access-list log-update-threshold 10000
```

crypto key master

This command creates, deletes, or specifies a master encryption key, which encodes all other keys on the XSR including AAA user database and private keys used by PKI (**user.dat**, **cert.dat** and **hostkey.dat**). Before configuring your VPN, you must generate this key.

Caution: The master encryption key is stored in hardware, not Flash, and you cannot read the key - only overwrite the old key by writing a new one. To ensure router security, it is critical not to compromise the key. There are situations where you may want to keep the key, for example, to save the user database off-line in order to later download it to the XSR. In order to encrypt the user database, you need the same master key, indicating the key designation with the **master key specify** command. Be aware that if the XSR is inoperable and you press the Default button, the master key is erased and you must generate a new one.

Syntax

crypto key master {**generate** | **remove** | **specify**}

generate	Create a master encryption key.
remove	Delete the master encryption and host key pair (*hostkey.dat*).
specify	Specify a master encryption key.

Mode

Global configuration: **XSR(config)#**

Sample Output

The following output displays when a master key is generated:

```
XSR(config)#crypto key master generate
New key is 8573 4583 3994 2ff5
           183b 4bdf fe92 dbc1
           1132 ffe0 f8d9 3759
```

A script displays when a master key is specified, prompting you for the following information:

```
XSR(config)#crypto key master specify
Specify first encryption key in hex digits:   []: 8573 4583 3994 2ff5
Specify second encryption key in hex digits:  []: 183b 4bdf fe92 dbc1
Specify third encryption key in hex digits:   []: 1132 ffe0 f9d9 3759
Are you sure?  [y]:
```

Crypto Map Mode Commands

crypto map (Global IPSec)

This command creates or modifies a crypto map entry. It also acquires Crypto Map mode. Along with the setting of a transform-set, this constitutes IPSec Phase 2 configuration.

In Crypto Map mode, the following sub-commands are available:

- **`match address`** - Correlates ACLs to map. Refer to page 14-111 for the command definition.
- **`mode`** - Selects encapsulation type - tunnel or transport- for a transform-set. Refer to page 14-112 for the command definition.
- **`set peer`** - Specifies peer's IP address. Refer to page 14-113 for the command definition.
- **`set security-association level per-host`** - Specifies separate SAs be requested for each source/destination host pair. Refer to page 14-114 for the command definition.
- **`set transform-set`** - Correlates transform-sets with map. Refer to page 14-114 for the command definition.

Crypto Map

Crypto *maps* provide two functions: filter and classify traffic to be protected as well as define the policy to be applied to that traffic. The first use affects the flow of traffic on an interface; the second affects the negotiation performed (via IKE) on behalf of that traffic.

IPSec crypto maps link definitions of the following:

- Which traffic should be protected.
- Which IPSec peers the protected traffic can be forwarded to - these are the peers with which a Security Association (SA) can be built.
- Which transform-sets are acceptable for use with the protected traffic.
- How keys and SAs should be used or managed.

Note: A crypto map has no effect until it is attached to an interface.

Crypto Map Rules

A crypto map is a collection of rules, each with a different *seq-num* but the same *map-name*. So, for a given interface, you can have certain traffic forwarded to one IPSec peer with specified security applied to that traffic, and other traffic forwarded to the same or a different IPSec peer with different IPSec security applied. To accomplish this you create two crypto maps, each with the same *map-name*, but each with a different *seq-num*. Crypto map rules are searched in order of *seq-num*. Sequence numbers, in addition to determining the order in which traffic is tested against the rules, are used as an anti-replay device to reject duplicate and old packets and so prevent an intruder from copying a conversation and using it to work out encryption algorithms.

Syntax

crypto map *map-name seq-num* [*ipsec-isakmp*]

map-name	Crypto map identification. This is the name assigned when the crypto map was created.
seq-num	32-bit digit you assign to the crypto map. Range: 1 to 4096.
ipsec-isakmp	This value provides backward compatibility with the industry-standard CLI. It is not mandatory.

Syntax of the "no" Form

To delete a crypto map entry, use the *no* form of this command:

no crypto map *map-name* [*seq-num*]

Mode

Global configuration: **XSR(config)#**

Next Mode

Crypto Map configuration: **XSR(config-crypto-m)#**

Sample Output

The following example creates the crypto map *ACMEmap*:

```
XSR(config)#crypto map ACMEmap 7
XSR(config-crypto-m)#set transform-set esp-3des-sha
XSR(config-crypto-m)#match address 120
```

match address

This command specifies an access control list (ACL) for a crypto map entry. An ACL is applied *bidirectionally* by IPSec and the XSR considers its "source" as the *local* address and its "destination" as the *remote* address so typically only *one* match address and ACL is needed to define traffic with a peer.

Syntax

match address [*access-list-id*]

access-list-id	Identifies the extended ACL by its number. This value should match the *access-list-number* argument of the ACL being matched.

Syntax of the "no" Form

Use the *no* form to remove the ACL from a crypto map entry:

no match address [*access-list-id*]

Default

No access lists are matched to the crypto map entry.

Mode

Crypto Map configuration: **XSR(config-crypto-m)#**

Example

The following static crypto map example shows the minimum required crypto map configuration when IKE will be used to establish the SAs:

```
XSR(config)#crypto map ACMEmap 7 ipsec-isakmp
XSR(config-crypto-m)#match address 101
XSR(config-crypto-m)#set transform-set my_t_set1
XSR(config-crypto-m)#set peer 10.0.0.1
```

mode

This command selects one of two IPSec-defined encapsulation modes, tunnel or transport, for a transform-set. Tunnel mode, the default, typically is used with VPNs because the entire private network packet is carried as the payload of the IPSec packet. Transport mode carries only the payload (TCP or UDP typically) of the private network packet as the payload of the IPSec packet.

Note: Transport mode *must* be selected for a Windows L2TP/IPSec client to operate properly.

Syntax

mode [*tunnel* | *transport*]

tunnel	Tunnel mode.
transport	Transport mode.

Syntax of the "no" Form

The *no* form of this command resets the mode to the default:

no mode

Default

Tunnel mode

Mode

Crypto Map configuration: **XSR(config-crypto-m)#**

Example

This example defines a transform-set and changes the mode to *transport* mode. The mode value only applies to IP traffic with source and destination addresses at the local and remote IPSec peers.

```
XSR(config)#crypto ipsec transform-set newer esp-des esp-sha-hmc
XSR(config)crypto map ACMEmap 14
XSR(config-crypto-m)#mode transport
```

set peer

This command specifies an IPSec peer in a crypto map entry. When traffic passing through the interface matches a crypto map entry, a tunnel is opened to the peer specified by this command.

Syntax

set peer *ip-address*

ip-address	Specifies the IPSec peer by its IP address.

Syntax of the “no” Form

To remove an IPSec peer from a crypto map entry, use the *no* form of this command:

no set peer {*hostname* | *ip-address*}

Default

No peer is defined

Mode

Crypto Map configuration: **XSR(config-crypto-m)#**

Example

This example shows a crypto map configuration when IKE is used to build Security Associations. In this example, an SA could be set up with either the IPSec peer at *10.0.0.1* or the peer at *10.0.0.2*.

```
XSR(config)#crypto map ACMEmap 7 ipsec-isakmp
XSR(config-crypto-m)#match address 101
XSR(config-crypto-m)#set transform-set my_t_set1
XSR(config-crypto-m)#set peer 10.0.0.1
```

set security-association level per-host

This command specifies that separate IPSec Security Associations (SAs) should be requested for each source/destination host pair.

Syntax

```
set security-association level per-host
```

Syntax of the “no” Form

The *no* form specifies that one SA should be requested for each crypto map ACL permit entry.

```
no set security-association level per-host
```

Default

For a given crypto map, all traffic between two IPSec peers matching a single crypto map ACL *permit* entry will share the same SA.

Mode

Crypto Map configuration: **XSR(config-crypto-m)#**

Example

The following example sets the SA request on a per-host basis:

```
XSR(config)crypto map ACMEmap
XSR(config-crypto-m)#set security-association level per-host
```

set transform-set

This command specifies which transform-sets can be used with the crypto map entry.

Syntax

set transform-set *transform-set-name1* [*transform-set-name2...transform-set-name6*]

transform-set-name	Name of the transform-set. Up to 6 can be specified.

Syntax of the “no” Form

The *no* form of this command removes all transform-sets from a crypto map entry:

```
no set transform-set
```

Mode

Crypto Map configuration: **XSR(config-crypto-m)#**

Example

This example defines two transform-sets, specifying both can be used within a crypto map entry. When traffic matches ACL *101*, the SA can use either transform-set *my_t_set1* (first priority) or *my_t_set2* (second priority) depending on which transform-set matches the remote peer's transform-sets.

```
XSR(config)#crypto ipsec transform-set my_t_set1 esp-des esp-sha-hmac
XSR(config)#crypto ipsec transform-set my_t_set2 ah-sha-hmac esp-des esp-sha-hmac
XSR(config)#crypto map ACMEmap 7 ipsec-isakmp
XSR(config-crypto-m)#match address 101
XSR(config-crypto-m)#set transform-set my_t_set1 my_t_set2
XSR(config-crypto-m)#set peer 10.0.0.1
```

Crypto Transform Mode Commands

crypto ipsec transform-set

This command defines a transform-set which is an acceptable combination of security protocols and algorithms to apply to IP Security protected traffic. During IPSec Security Association (SA) negotiation, peers agree to use a particular transform-set when protecting a particular data flow.

This command acquires Crypto Transform configuration Mode. The following sub-commands are available in this mode:

- **`set pfs`** - Specifies that IPSec should ask for PFS when seeking new SAs for this crypto map entry, or that IPSec requires PFS when getting requests for new SAs. Refer to page 14-116 for the command definition.
- **`set security-association lifetime`** - Specifies the interval used when negotiating IPSec SAs. Refer to page 14-117 for the command definition.

A transform-set is an acceptable combination of security protocols, algorithms and other settings to apply to IP Security-protected traffic. During IPSec SA negotiation, the peers agree to use a particular transform-set when protecting a particular data flow.

Syntax

`crypto ipsec transform-set` *`transform-set-name transform1 [transform2 [transform3]]`*

`transform-set-name`	Name of the transform-set to create or modify.
`transform1`	Specify up to 3 transforms defining the IPSec security protocols and algorithms. The choices are: • *ah-md5-hmac*: AH transform with HMAC-MD5 algorithm. • *ah-sha-hmac*: AH transform with HMAC-SHA algorithm. • *esp-3des*: ESP transform with 56-bit DES encryption (168-bits). • *esp-aes*: ESP transform with 128-bit AES encryption. • *esp-des*: ESP transform with 168-bit Triple DES encryption. • *esp-md5-hmac*: ESP transform with HMAC-MD5 data integrity algorithm. • *esp-null*: ESP transform with no encryption. • *esp-sha-hmac*: ESP transform with HMAC-SHA data integrity algorithm.

Mode of the “no” Form

The *no* form of the command deletes a transform-set:

no crypto ipsec transform-set *transform-set-name*

Mode

Global configuration: **XSR(config)#**

Next Mode

Crypto Transform configuration: **XSR(cfg-crypto-tran)#**

Example

The following example defines the transforms to apply for *t-set1* SA negoatiation:

```
XSR(config)#crypto ipsec transform-set t-set1 esp-3des esp-sha-hmac
```

set pfs

This command specifies that IPSec ask for Perfect Forward Secrecy (PFS) when requesting new Security Associations (SAs) for this crypto map entry, or that IPSec requires PFS when receiving requests for new SAs.

PFS is a security condition under which there is confidence that the compromise of a session’s key will not lead to easier compromise of the key used in the next session (after the key is refreshed). When PFS is used a session’s keys are generated independently, so a key compromised in one session will not affect the keys used in subsequent sessions.

Note: Due to the lack of an IETF standard, IKE Diffie-Helman bit groups 2048, 3072, and 4096 are not enabled.

Syntax

set pfs [*group1* | *group2*]

group1	Specifies that IPSec should use the 768-bit Diffie-Hellman prime modulus group when performing the new Diffie-Hellman exchange.
group2	Specifies that IPSec should use the 1024-bit Diffie-Hellman prime modulus group when performing the new Diffie-Hellman exchange.

Syntax of the “no” Form

Use the *no* form of the command for IPSec *not* to request PFS:

no set pfs

Default

Disabled

Mode

Crypto Transform configuration: **XSR(cfg-crypto-tran)#**

Example

This example selects PFS group 2 whenever a new SA is negotiated for crypto map *ACMEmap*:

```
XSR(config)#crypto map ACMEmap 7 ipsec-isakmp
XSR(config)#crypto ipsec transform-set t-set1 esp-3des esp-sha-hmac
XSR(cfg-crypto-tran)#set pfs group2
```

set security-association lifetime

This command sets the lifetime interval used when negotiating IPSec Security Associations (SAs). Data passing through the XSR is encrypted using keys generated during IKE exchange. The lifetime of those keys may be defined in seconds or in data volume which was encrypted using those keys. When that lifetime expires new keys are generated and traffic continues to be passed using new keys.

Syntax

set security-association lifetime {**seconds** *seconds* | **kilobytes** *kilobytes*}

seconds	The interval an SA lives before expiring, ranging from 300 to 86,400,000 seconds.
kilobytes	The volume of traffic, in KBytes, that can pass between IPSec peers using a given SA before that SA expires, ranging from 1 MByte to 1000 GBytes.

Syntax of the “no” Form

The *no* form of this command disables the specified lifetime metric. It does not reset the default:

no set security-association lifetime {**seconds** | **kilobytes**}

Default

3600 seconds with no limit on traffic volume.

Mode

Crypto Transform configuration: **XSR(cfg-crypto-tran)#**

Example

The following example sets the SA lifetime to *7,200 KBytes* and disables the *seconds* parameter:

```
XSR(cfg-crypto-tran)#)#set security-association lifetime kilobytes 7200
XSR(cfg-crypto-tran)#)#no set security-association lifetime seconds
```

Crypto Show Commands

show crypto ipsec sa

This command displays current Security Associations (SAs) settings.

Syntax

show crypto ipsec sa [**map** *map-name* | **address**]

map-name	Shows any existing SAs created for the crypto map set named *map-name.*
address	Shows all existing SAs, sorted by the destination address (either the local address or the address of the IPSec remote peer) and then by protocol (AH or ESP).

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following is sample output when NAT is *not* present between the crypto endpoints. The first section is the inbound SA, and the second section, the outbound SA. The UDP port follow the the IP address for crypto endpoints when a NAT is present.

```
XSR#show crypto ipsec sa
10.1.1.2/32, UDP, 1701  ==>  10.2.1.34/32, UDP, 1701 : 71 packets
ESP: SPI=f5ae2b52, Transform=3DES/HMAC-SHA, Life=3575S/249929KB
Local crypto endpt.=10.2.1.34, Remote crypto endpt.=10.1.1.2
Encapsulation=Transport
10.2.1.34/32, UDP, 1701  ==>  10.1.1.2/32, UDP, 1701 : 36 packets
ESP: SPI=5419ec15, Transform=3DES/HMAC-SHA, Life=3575S/249933KB
Local crypto endpt.=10.2.1.34, Remote crypto endpt.=10.1.1.2
Encapsulation=Transport
```

The following is sample output when NAT *is* present between the crypto endpoints. Note that UDP-Encaps displays, indicating that encapsulation is enabled with a NAT present.

```
10.2.1.10/32, UDP, 1701  ==>  10.2.1.34/32, UDP, 1701 : 52 packets
ESP: SPI=40d5e065, Transform=3DES/HMAC-SHA, Life=3589S/249932KB
Local crypto endpt.=10.2.1.34:4500, Remote crypto endpt.=10.2.1.10:41108
Encapsulation=Transport UDP-Encaps
10.2.1.34/32, UDP, 1701  ==>  10.2.1.10/32, UDP, 1701 : 32 packets
ESP: SPI=5c0f6fb5, Transform=3DES/HMAC-SHA, Life=3589S/249934KB
Local crypto endpt.=10.2.1.34:4500, Remote crypto endpt.=10.2.1.10:41108
Encapsulation=Transport UDP-Encaps
```

Parameter Description

10.2.1.10/32, UDP, 1701	IP address, protocol, and protocol port number of the *source* ACL entry associated with this SA.
10.2.1.34/32, UDP, 1701	IP address, protocol, and protocol port number of the *destination* ACL entry associated with this SA.
52 packets	Number of packets processed by this SA.

ESP	Type of SA: either ESP or AH.
SPI=40d5e065	Unique Security Parameter Index (SPI) number for the SA.
Transform	Encryption algorithm set.
Life=3589s/249932KB	Lifetime of the SA in seconds and KBytes.
Local crypto endpt.-10.2.1.34:4500	IP address and port number of the local crypto peer.
Remote crypto endpt.-10.2.1.34:4500	IP address and port number of the remote crypto peer.
Encapsulation	ESP or AH Encoding Mode.
UDP-Encaps	Indicates NAT is present between the crypto endpoints.

show crypto ipsec transform-set

This command displays configured transform-sets. IPSec transform-sets created with EZ-IPSec configuration are marked with an asterisk (*) in the **show** output. These proposals may not be used in other user-defined IPSec policies. They are reserved for EZ-IPSec

Syntax

show crypto ipsec transform-set [*transform-set-name*]

transform-set-name	Shows transform-sets with the specific *transform-set-name* only.

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following example was produced from *manually* configured transform-sets:

```
XSR#show crypto ipsec transform-set
Name                          PFS         ESP       ESP-AH      AH          IPCOMP
esp-3des-md5                  Disabled    AES       HMAC-MD5    None        None
ah-sha                        Disabled    None      None        HMAC-SHA    None
```

The following output was produced by *EZ-IPSec* transform-sets:

```
XSR#show crypto ipsec transform-set
Name                          PFS         ESP       ESP-AH      AH          IPCOMP
*ez-esp-3des-sha-pfs          Modp768     3DES      HMAC-SHA    None        None
*ez-esp-3des-sha-no-pfs       Disabled    3DES      HMAC-SHA    None        None
*ez-esp-3des-md5-pfs          Modp768     3DES      HMAC-MD5    None        None
*ez-esp-3des-md5-no-pfs       Disabled    3DES      HMAC-MD5    None        None
*ez-esp-aes-sha-pfs           Modp768     AES       HMAC-SHA    None        None
*ez-esp-aes-sha-no-pfs        Disabled    AES       HMAC-SHA    None        None
*ez-esp-aes-md5-pfs           Modp768     AES       HMAC-MD5    None        None
*ez-esp-aes-md5-no-pfs        Disabled    AES       HMAC-MD5    None        None
```

show crypto map

This command displays the crypto map configuration. IPSec crypto maps created with EZ-IPSec configuration are marked with an asterisk (*) in the leftmost column of the **show** output. These proposals may not be used in other user-defined IPSec policies. They are reserved for EZ-IPSec.

Syntax

show crypto map [**interface** *type* | **tag** *map-name*]

type	Shows only the crypto map set applied to the specified interface including: *ATM, BRI, Dialer, Fast/GigabitEthernet, Multilink,* or *Serial.*
map-name	Shows only the *crypto map* set with the specified *map-name*.

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

```
XSR#show crypto map
Crypto Map Table
Name         Policy rule list
ezipsec      n03;c03
test         test.10;test.20

IPSec Policy Rule Table
Name   ACL    Disp      Mode   Bundle Gateway          Proposals
*c03     c03   Process   Tunnel SPD    141.154.196.87  ez-esp-3des-sha-pfs
                                                       ez-esp-3des-md5-pfs
                                                       ez-esp-aes-sha-pfs
                                                       ez-esp-aes-md5-pfs
                                                       ez-esp-3des-sha-no-pfs
                                                       ez-esp-3des-md5-no-pfs
                                                       ez-esp-aes-sha-no-pfs
                                                       ez-esp-aes-md5-no-pfs
*n03   n03    Process   Tunnel SPD    141.154.196.87  ez-esp-3des-sha-pfs
                                                       ez-esp-3des-md5-pfs
                                                       ez-esp-aes-sha-pfs
                                                       ez-esp-aes-md5-pfs
                                                       ez-esp-3des-sha-no-pfs
                                                       ez-esp-3des-md5-no-pfs
                                                       ez-esp-aes-sha-no-pfs
                                                       ez-esp-aes-md5-no-pfs
test.10 110    Process   Trans  SPD    0.0.0.0         T/Med ah-sha
test.20 120    Process   Tunnel SPD    1.1.2.1         T/Med esp-3des-md5

EZ-IPSec Access Control List

Name Local Address        Remote Address       Prot       Lport   Rport
*c03  10.120.122.17        0.0.0.0/0            ANY        0       0
*n03  172.16.19.0/24       0.0.0.0/0            ANY        0
```

Interface CLI Commands

crypto map

This command applies a previously defined *crypto map* to an interface. It is governed by the following rules:

- A crypto map must be assigned to an interface before that port can provide IPSec services.
- Only 1 crypto map can be assigned an interface although it can be attached to multiple ports.
- A crypto map may not be assigned to an interface that already has **crypto ezipsec** enabled.
- Crypto maps may not be assigned to a VPN interface (it is invalid at Interface VPN mode).

Syntax

crypto map *map-name*

map-name	*Crypto map* ID assigned when the crypto map was created.

Syntax of the “no” Form

Delete a crypto map from the interface with the *no* form of this command:

no crypto map [*map-name*]

Mode

Interface configuration: **XSR(config-if<xx>)#**

Next Mode

Crypto Map configuration: **XSR(config-crypto-m)#**

Sample Output

This example assigns crypto map *ACMEmap* to the *F1* interface. When traffic passes through *F1*, it will be evaluated against all the crypto map entries in the *ACMEmap* set. When outbound traffic matches an access list in one of the *ACMEmap* crypto map entries, a Security Association will be established for that crypto map entry's configuration (if no SA or connection already exists).

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#crypto map ACMEmap
```

crypto ezipsec

This command creates a suite of IPSec policies, sorted by cryptographic strength, that are offered to the remote security gateway. The gateway selects one of these policies based on its local configuration. EZ-IPSec relies upon the IKE Mode Configuration protocol to obtain an IP address from the remote security gateway.

An EZ-IPSec crypto map is also created and attached to the interface under configuration. Refer to the *XSR User's Guide* for specific examples and how **`crypto ezipsec`** is used with RIP and NAT. Be aware of the following rules governing this command:

- **`Crypto ezipsec`** may not be enabled on an interface that already has a crypto map.
- Crypto maps may be attached to *other* network interfaces.
- EZ-IPSec parameters cannot be changed but can be supplemented with custom values.

Syntax

```
crypto ezipsec
```

Syntax of the "no" Form

```
no crypto ezipsec
```

Default

Disabled

Mode

Interface configuration: **`XSR(config-if<xx>)#`**

Example

The following example configures EZ-IPSec on Serial interface 1:

```
XSR(config-if<S1/0>)#crypto ezipsec
```

Interface VPN Commands

interface vpn

This command acquires virtual Interface VPN configuration mode from which you can configure the following sub-commands:

- **`copy-tos`** - Copies TOS bits during the encapsulation/decapsulation process. Refer to page 14-124 for the command definition.
- **`description`** – Describes the VPN interface. Refer to page 14-125 for the command definition.
- **`ip address negotiated`** – Requires a site-to-site tunnel to obtain an IP address from the remote tunnel gateway via PPP or IKE Mode Config. Refer to page 14-126 for the command definition.

- **`ip multicast-redirect -`** Native IPSec tunnels attached to VPN interfaces will not easily forward multicast traffic multicast packet redirection to the unicast address of the remote tunnel endpoint. Refer to page 14-126 for the command definition.
- **`ip address`** - Defines an explicit IP address on this virtual interface. Refer to page 5-151 for the command description.
- **`ip nat source`** - Controls NAT on packets entering this VPN port. Refer to page 5-186 for the command description.
- **`ip rip`** commands - Configures RIP options on the VPN interface. Refer to the "Configuring the Internet Protocol" on page 5-83 chapter for descriptions of RIP commands.
- **`ip split-horizon`** - Sets RIP split-horizon options on the VPN port. Refer to page 5-130 for the command description.
- **`ip unnumbered`** - Creates an unnumbered VPN interface. Refer to page 5-166 for the command description.
- **`service-policy`** - Attaches a policy map to an VPN output or input interface. Refer to page 14-127 for the command description.
- **`tunnel`** - Creates a tunnel to a VPN gateway. Refer to page 14-127 for the command description.

Some VPN configuration properties are associated with a specific network interface or require creation of virtual network interfaces that represent tunnels.

This section defines the VPN-related subcommands provided by the **`interface vpn`** command.

A VPN interface is a special form of a virtual network interface that represents an IPSec tunnel with EZ-IPSec automatic configuration, L2TP, or PPTP tunnel(s). It is required to support VPN tunnels which have IP addresses. These tunnels should not be confused with tunnel mode in IPSec. A tunnel on a VPN interface has IP addresses at both ends and is used by the routing subsystem like any other network interface.

A VPN interface can be configured as follows:

- **`interface vpn 4 point-to-point`**
- **`interface vpn 3 multi-point`**

Point-to-Point interfaces are used when defining an outbound tunnel to another gateway. This interface type, in conjunction with the **`tunnel`** command, is suited to initiating outbound tunnels to other security gateways that support dynamic IP address assignment.

Note: The **`tunnel`** command is a sub-command of **`interface vpn.`**

Each outbound tunnel is associated with a VPN interface. That interface, which can be configured into the routing protocols, is considered down until the tunnel has connected and an IP address has been obtained from the remote VPN gateway.

Note: Only one tunnel may be defined per point-to-point VPN interface.

A *multi-point* interface accepts many inbound tunnels and is used when the XSR is configured as a remote access VPN gateway.

Note: The **no shutdown** command is not required to bring up the virtual interface because it is always enabled.

Syntax

interface vpn {**number**}{**point-to-point** | **multi-point**}

number	VPN interface number ranging from 1 to 255.
point-to-point	VPN port type *initiating outbound* tunnels to another gateway.
multi-point	VPN port type terminating *inbound* tunnels from a remote access VPN gateway.

Syntax of the "no" Form

The following command deletes the specified VPN interface:

no interface vpn

Mode

Global configuration: **XSR(config)#**

Next Mode

Interface configuration: **XSR(config-int-vpn)#**

Example

The following example creates VPN interface 57:

```
XSR(config)#interface vpn 57
XSR(config-int-vpn)#
```

copy-tos

This command copies TOS bits during the encapsulation/decapsulation process. It can be applied to a VPN interface or inserted in the crypto isamp peer command. When applied, the command copies the TOS byte from the inner to the outer header for output packets. For input packets, it copies the TOS byte from the outer to the inner header.

Syntax

copy-tos

Syntax of the "no" Form

The following no form of the command removes the TOS copy action:

no copy-tos

Mode

VPN Interface configuration: XSR(config-if<xx>)#

Example

The following example configures VPN interface 1 with an IP address, and TOS copy enabled. It also sets a peer IP address, GRE, and turns on the associated VPN tunnel.

```
XSR(config)#interface vpn 1
XSR(config-int-vpn)#ip address 20.20.20.1/24
XSR(config-int-vpn)#copy-tos
XSR(config-int-vpn)#service-policy output vpn
XSR(config-int-vpn)#tunnel t1
XSR#(config-tms-tunnel)#set protocol gre
XSR#(config-tms-tunnel)#set peer 10.10.10.2
XSR#(config-tms-tunnel)#set active
XSR#(config-tms-tunnel)#no shutdown
```

description

This commands describes a VPN interface and any tunnel it contains.

Syntax

description *comment*

comment	Everything to the end of the line is recorded as a comment. Use quotation marks for multiple words.

Syntax of the “no” Form

The *no* form of this command deletes the description described earlier:

```
no description
```

Mode

Interface Internet Protocol configuration: **XSR(config-int<vpn>)#**

Example

The following example describes *ACME_VPN*:

```
XSR(config)#interface vpn 57 multi-point
XSR(config-int<vpn>)#description ACME_VPN
```

ip address negotiated

This command marks the VPN interface to dynamically get its IP address via the tunnel protocol. PPTP and L2TP protocols use PPP IPCP and IPSec/IKE uses the Mode Configuration protocol.

Syntax

```
ip address negotiated
```

Syntax of the “no” Form

```
no ip address negotiated
```

Mode

Interface Internet Protocol configuration: **XSR(config-int<vpn>)#**

Example

The following example sets the VPN interface to get its IP address from the tunnel protocol:

```
XSR(config)#interface vpn 57 point-to-point
XSR(config-int<vpn>)#ip address negotiated
```

ip multicast-redirect

This command controls redirection of multicast packets to the unicast address of the remote tunnel endpoint or to an explicitly defined address such as another IP address at the end of an unnumbered tunnel. The command is useful because native IPSec tunnels attached to VPN interfaces will not easily forward multicast traffic without substantial crypto map configuration.

Multicast redirection *must* be enabled to support RIP over IPSec tunnels when explicit multicast policy rules are not included in the Security Policy Database. Redirection is not required for PPTP and L2TP tunnels.

Note: Multicast redirection, if enabled, applies to all tunnels terminating at a point-to-multipoint VPN interface.

Syntax

ip multicast-redirect [*tunnel-endpoint* | *ip-address*]

tunnel-endpoint	Redirects multicast to the remote tunnel endpoint's IP address as dynamically set during tunnel creation.
ip-address	Redirects multicast traffic to an explicit, predefined address.

Syntax of the “no” Form

The *no* form of the command disables multicast packet redirection and allows multicast traffic to flow through the tunnel without modification:

no ip multicast-redirect [*tunnel-endpoint* | *ip-address*]

Mode

Internet Protocol Interface configuration: **`XSR(config-int<vpn>)#`**

Example

This example redirects multicast traffic to the remote tunnel server:

```
XSR(config)#interface vpn 57 multi-point
XSR(config-int<vpn>)#ip multicast-redirect tunnel-endpoint
```

service-policy

This command attaches a policy map to an VPN output or input interface. You can attach a single policy map to one or more interfaces.

Syntax

`service-policy` [*input | output*] *policy-map-name*

policy-map-name	Attaches the specified policy map onto the output port.

Syntax of the "no" Form

The *no* form of the command removes a policy map from the interface:

`no service-policy` [*input | output*]

Mode

Interface configuration: **`XSR(config-if<xx>)#`**

Example

The following example attaches service policy VPNpolicy to VPN output interface 1:

```
XSR(config)#interface vpn 1
XSR(config-int<vpn>)#service-policy output VPNpolicy
```

Tunnel Commands

tunnel

This sub-command of **`interface vpn`** names a tunnel created at boot time that links this VPN interface with another VPN gateway. The VPN interface, with its tunnel, is equivalent to a point-to-point interface. Issuing the command acquires Tunnel configuration mode, making available the following sub-commands:

- **`set active`** - Enables the VPN tunnel. Refer to page 14-128 for the command definition.
- **`set heartbeat`** - Monitors tunnel connectivity. Refer to page 14-129 for the command definition.
- **`set peer`** - Specifies the physical IP address of the remote VPN gateway. Refer to page 14-130 for the command definition.

- **set protocol -** Defines the VPN tunneling protocol used when the tunnel is created: client mode or network extension mode. Refer to page 14-130 for the command definition.
- **set user -** Username employed when connecting to the remote peer. Refer to page 14-131 for the command definition.

Syntax

tunnel *tunnel-name*

tunnel-name	The name assigned to the tunnel.

Syntax of the "no" Form

The *no* form of this command deletes the tunnel:

no tunnel *tunnel-name*

Mode

Interface Internet Protocol configuration: **XSR(config-int-vpn)#**

Next Mode

Tunnel configuration: **XSR#(config-tms-tunnel)#**

Example

The following example adds the tunnel *ACME_VPN:*

```
XSR(config)#interface vpn 57 multi-point
XSR(config-int<vpn>)#tunnel ACME_VPN
XSR#(config-tms-tunnel)#
```

set active

This command enables the tunnel.

Syntax

set active

Syntax of the "no" Form

The *no* form of this command disables the tunnel:

no set active

Default

Enabled

Mode

Tunnel configuration: **XSR(config-tms-tunnel)#**

Example

The following example enables the tunnel *ACME_VPN:*

```
XSR(config)#interface vpn 57 multi-point
XSR(config-int<vpn>)#tunnel ACME_VPN
XSR#(config-tms-tunnel)#set active
```

set heartbeat

This command configures the mechanism to probe a tunnel peer to monitor tunnel connectivity. Ping is used over IKE/IPSec tunnels configured with dynamically assigned addresses.

Syntax

set heartbeat {*interval | retries*>} [*A.B.C.D*]

interval	Interval between heartbeat tries before timing out, ranging from 1 to 3600 seconds. Zero (0) disables the heartbeat.
retries	Number of retries before the tunnel is declared down, ranging from 3 to 100.
A.B.C.D.	IP address of a specified remote peer to ping to monitor tunnel connectivity.

Syntax of the “no” Form

The *no* form of this command disables the heartbeat:

```
no set heartbeat
```

Defaults

- Interval: 6 seconds
- Retries: 3

Mode

Tunnel configuration: **XSR#(config-tms-tunnel)#**

Example

The following example sets tunnel heartbeat values:

```
XSR(config)#interface vpn 57 multi-point
XSR(config-int<vpn>)#tunnel “ACME VPN”
XSR#(config-tms-tunnel)#set heartbeat 50 10 192.168.57.9
```

set peer

This command specifies the physical IP address of the remote VPN gateway.

Syntax

set peer *ip-address*

ip-address	IP address of the peer.

Syntax of the “no” Form

no set peer *ip-address*

Mode

Tunnel configuration: **XSR#(config-tms-tunnel)#**

Example

The following example sets the IP address of the remote VPN gateway:

```
XSR(config)#interface vpn 57 multi-point
XSR(config-int<vpn>)#tunnel ACME_VPN
XSR#(config-tms-tunnel)#set peer ip-address 192.168.57.9
```

set protocol

This command defines the VPN tunneling protocol - Generic Routing Encapsulation (GRE) or IP Security (IPSec) - used to create the tunnel.

IPSec accepts one of two sub-commands that create a Client or Network Extension mode site-to-site tunnel. *Client mode* creates NAT on the VPN interface to hide the addresses of the trusted network (attached to *F1*). IPSec security policy encrypts data passing to and from the IP address assigned to the tunnel. *Network extension mode* creates IPSec security policies that encrypt traffic flowing to the trusted network via the tunnel in addition to securing traffic flowing to the tunnel's assigned address.

Syntax

set protocol {**gre** | **ipsec**}[*client-mode* | *network-extension-mode*]

gre	GRE tunneling protocol.
ipsec	IPSec tunneling protocol.
client-mode	Initiates a Client-mode EZ-IPSec tunnel.
network-extension-mode	Initiates a NEM EZ-IPSec tunnel.

Syntax of the “no” Form

The *no* form of this command negates the protocol selected earlier:

no set protocol

Mode

Tunnel configuration: **XSR#(config-tms-tunnel)#**

Default

IPSec

Examples

The following example sets the IPSec tunnel protocol in *client* mode:

```
XSR(config)#interface vpn 29 point-to-point
XSR(config-int<vpn>)#tunnel ACME_VPN
XSR#(config-tms-tunnel)#set protocol ipsec client-mode
```

The example below connects a GRE tunnel attached to a VPN interface:

```
XSR(config)#interface vpn 2 point-to-point
XSR(config-int<vpn>)#ip address 192.168.1.123 255.255.255.0
XSR#(config-int<vpn>)#tunnel my-gre-tunnel
XSR#(config-tms-tunnel)#set protocol gre
XSR#(config-tms-tunnel)#set peer 10.1.2.3
XSR#(config-tms-tunnel)#set active
```

set user

This command specifies a user's identity when connecting to a peer. It invokes EZ-IPSec by applying the credentials (password and/or certificate) used during tunnel creation obtained from the AAA subsystem. An EZ-IPSec tunnel uses *aggressive* mode with the username as the IKE identity. Refer to the **aaa user**, **user-id**, and **show crypto ca certificate** commands for more information.

Syntax

set user *username*

username	Username employed when connecting to the peer.

Mode

Tunnel configuration: **XSR#(config-tms-tunnel)#**

Examples

The following example specifies the pre-shared key of a peer by *username*:

```
XSR(config)#interface vpn 29 point-to-point
XSR(config-int<vpn>)#tunnel ACME_VPN
XSR#(config-tms-tunnel)#set user jonathan
```

The following example specifies the pre-shared key of a peer by *certificate*:

```
XSR(config)#interface vpn 29 point-to-point
XSR(config-int<vpn>)#tunnel ACME_VPN
XSR#(config-tms-tunnel)#set user certificate
```

Tunnel Clear and Show Commands

clear tunnel

This command terminates a non-GRE tunnel associated with a user or tunnel ID. Tunnels will re-establish themselves if set to do so unless the user is disabled in its database. For example, a cleared IPSec tunnel will re-establish if traffic is initiated.

Note: This command terminates all but GRE and GRE/IPSec tunnels with an error message displayed if you attempt to do so. To bring down a GRE tunnel, remove its interface or use the **no set active** command.

L2TP and PPTP tunnels will be disconnected on the server side. The client side of the tunnel will time out after its designated timeout period.

Syntax

clear tunnel *<user-ID | <tunnel-ID>*

user-ID	Name of the VPN user.
tunnel-ID	Identification number associated with this tunnel.

Mode

Privileged EXEC: **XSR#**

Example

The following example terminates tunnel *40000001*:

```
XSR#clear tunnel 40000001
```

show tunnels

This command lists all tunnels currently connected to the XSR.

Syntax

show tunnels *<user-ID | tunnel-ID>*

user-ID	Name of the VPN user.
tunnel-ID	Identification number associated with this tunnel.

Mode

Privileged EXEC: **XSR#**

Sample Output

The following is sample output queried by the *xsrclient* User-ID:

```
XSR#show tunnels xsrclient
```

```
User: xsrclient

          Tunnel ID:                  40000001
          VPN Interface:              VPN1
          Group:                      xsrgroup
          Connect Time:               11/05/2003, 23:39
          Protocol:                   L2TP
          Authentication Method:      MS-CHAPv2
          Packets In/Out:             0000000088/0000000027
          Errors In/Out:              0000000000/0000000000
          Discards In/Out:            0000000000/0000000000
```

The following is sample output queried by the Tunnel ID *40000001*:

```
XSR#show tunnel 40000001

Tunnel ID: 40000001

          User:                       xsrclient
          VPN Interface:              VPN1
          Group:                      xsrgroup
          Connect Time:               11/05/2003, 23:39
          Protocol:                   L2TP
          Authentication Method:      MS-CHAPv2
          Packets In/Out:             0000000088/0000000027
          Errors In/Out:              0000000000/0000000000
          Discards In/Out:            0000000000/0000000000
```

Parameter Description

VPN Interface	VPN port number to which the client is connected.
User ID	Name of the VPN user.
Tunnel ID	Tunnel identification number associated with this tunnel.
Group ID	VPN group name (if authenticated through AAA)
Connect Time	Start time and date for the connection.
Protocol Type	Type of protocol used in relation to this tunnel (e.g. PPTP, GRE, IPSec).
Authentication Method	Method of authentication (shared key/certificate, MS-CHAP, etc.)
Packets In/Out	Sum of incoming and outgoing packets.
Errors In/Out	Sum of incoming and outgoing packets with errors.
Discards In/Out	Sum of discarded incoming and outgoing packets.

Additional Tunnel Termination Commands

ip local pool

This command configures a local pool of IP addresses for when a remote peer connects to a point-to-multipoint interface or for use by DHCP.

Note: If an **aaa user** is configured to use a static IP address which belongs to a local IP pool, you must exclude that address from the local pool to prevent it from being assigned to another user.

The command acquires IP Local Pool configuration mode and provides these sub-commands:

- **exclude** - Bars a range of IP addresses from the local pool. Refer to page 14-135 for the sub-command definition.
- **exit** - Quits IP Local Pool configuration mode. Refer to page 14-135 for the sub-command definition.

Syntax

ip local pool *pool-name IP-address subnet*-mask

pool-name	Name of a particular local address pool.
IP-address	Base address of an IP subnet used to allocate IP addresses.
subnet-mask	Mask of that IP subnet. All subnet address bits matching zero bits in the mask must also be zero; that is, subnet and mask must be zero. May be expressed as **A.B.C.D** or **/<0-32>**.

Note: The pool size (mask) must be **/16** or higher (Class B or C) thus limiting any one pool to 64,000 IP addresses.

Syntax of the “no” Form

Use the *no* form of this command to delete an IP address from the pool:

no ip local pool *pool-name*

Mode

Global configuration: **XSR(config)#**

Next Mode

IP Local Pool configuration: **XSR(ip-local-pool)#**

Example

The following example creates a local IP address pool named marketing, which contains all IP addresses in the range *203.57.99.0* to *203.57.99.255*:

```
XSR(config)#ip local pool marketing 203.57.99.0 255.255.255.0
```

exclude

This sub-command bars the use of a range of IP addresses from an earlier created IP pool.

Syntax

exclude {*ip address*} {*number*}

ip address	Starting address to be excluded from pool.
number	Number of addresses to exclude, ranging from 1 to 65535.

Syntax of the “no” Form

The *no* form of this command removes the specified IP address from the exclude list:

exclude {*ip address*}{*number*}

Mode

Local IP Pool configuration: **XSR(ip-local-pool)#**

Examples

The following example excludes the *10* IP addresses between *192.168.57.100* and *192.168.57.110 f*rom local pool *HQ*:

```
XSR(config)#ip local pool HQ 192.168.57.0 255.255.255.0
XSR(ip-local-pool)#exclude 192.168.57.100 10
```

The following example negates the exclusion of IP addresses *192.168.57.105* and *192.168.57.106* from the earlier excluded range of IP addresses in local pool *HQ*:

```
XSR(config)#ip local pool HQ
XSR(ip-local-pool)#no exclude 192.168.57.105 2
```

exit

This sub-command quits IP Local Pool configuration mode.

Syntax

exit

Mode

IP Local Pool configuration: **XSR(ip-local-pool)#**

show ip local pool

This command displays statistics for any defined IP address pools.

Syntax

show ip local pool [*name*]

name	Name you specified for an IP address pool.

Mode

Privileged EXEC: **XSR#**

Sample Output

This output displays when the command is specified without a name:

```
XSR#show ip local pool

-----------IP Pools Statistics-----------
Pool    Subnet           Mask                  Free   In use   Excluded   Reserved
test    10.120.122.0     255.255.255.192 26    7      0        2
local   1.1.1.0          255.255.255.0         255    0        0          1
ddd     1.2.3.4          255.255.255.255       1      0        0          0
test    192.168.57.1     255.255.255.255       1      0        0          0
test1   192.168.57.252   255.255.255.255       1      0        0          0
test3   192.168.58.0     255.255.255.0         246    0        10         0
```

The following output displays when the command is specified with the name *test*:

```
XSR#show ip local pool test

-----------IP Pools Statistics-----------
Statistics of IP pool test
Available addresses:
        10.120.122.1
        10.120.122.2
        10.120.122.3
        10.120.122.5
        10.120.122.6
        10.120.122.7
        10.120.122.8
        10.120.122.9
        10.120.122.11
        10.120.122.12
        10.120.122.13
        10.120.122.14
        10.120.122.15
        10.120.122.16
        10.120.122.17
        10.120.122.18
        10.120.122.19
        10.120.122.20
```

```
        10.120.122.22
        10.120.122.24
        10.120.122.25
        10.120.122.26
        10.120.122.28
        10.120.122.31
        10.120.122.32
Inuse addresses:
        10.120.122.10
        10.120.122.21
        10.120.122.23
        10.120.122.27
        10.120.122.29
        10.120.122.30
        10.120.122.34
Excluded addresses:
Reserved addresses:
        10.120.122.0
        10.120.122.4
```

Parameter Description

Pool	Name of the IP pool.
Subnet	Mask of the IP pool.
Mask	IP address subnetwork of the IP pool.
Free	Sum of unused IP addresses within the pool.
In use	Sum of occupied IP addresses within the pool.
Excluded	Sum of IP addresses barred from use within the pool.
Reserved	Sum of IP addresses set aside within the pool, such as the initial address 192.168.57.0 within the 192.168.57.256 range.

DF Bit Commands

crypto ipsec df-bit (Global configuration)

This command sets the DF bit for the encapsulating header in VPN Tunnel Mode to *all* interfaces.

The *clear* setting for the DF bit should be used for encapsulating Tunnel Mode IPSec traffic when you can transmit packets larger than the available MTU size or you do not know the available MTU size.

Syntax

crypto ipsec df-bit {*clear* | *set* | *copy*}

clear	XSR will clear the DF bit from the outer IP header; the router may fragment the packet to add IPSec encapsulation.
set	XSR will set the DF bit in the outer IP header but the router may fragment the packet if the original packet had the DF bit cleared.

copy	XSR will search the original packet for the outer DF bit setting.

Defaults

- Disabled
- *Copy* setting

Mode

Global configuration: **XSR(config)#**

Example

The following example clears the DF bit on *all* interfaces:

```
XSR(config)#crypto ipsec df-bit clear
```

crypto ipsec df-bit (Interface configuration)

This command sets the DF bit for the encapsulating header in VPN Tunnel Mode to a *specific* interface.

The *clear* setting for the DF bit should be used for encapsulating Tunnel Mode IPSec traffic when you can transmit packets larger than the available MTU size or you do not know the available MTU size.

Note: This command *overrides* any existing DF bit global settings.

Syntax

crypto ipsec df-bit {*clear* | *set* | *copy*}

clear	XSR will clear the DF bit from the outer IP header; the router may fragment the packet to add IPSec encapsulation.
set	XSR will set the DF bit in the outer IP header but the router may fragment the packet if the original packet had the DF bit cleared.
copy	XSR will search the original packet for the outer DF bit setting.

Defaults

- Disabled
- *Copy* setting

Mode

Interface configuration: **XSR(config-if<xx>)#**

Example

The following example sets the DF bit on *F1*:

```
XSR(config-if<F1>)#crypto ipsec df-bit set
```

15

Configuring DHCP

Observing Syntax and Conventions

The CLI command syntax and conventions use the notation described in the following table.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[x]	[ ] Square brackets indicate an optional parameter (*italic*)
[x | y | z]	[|] Square brackets with vertical bar indicate a choice of values
{x | y | z}	{ | } Braces with vertical bar indicate a choice of a required value
[x {y | z}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<*xx*>)	*xx* signifies the interface type and number, class map, policy map or other value you specify; e.g., **F1**, **G3**, **S2/1.0**, **<Your Name>**. **F** indicates a FastEthernet, and **G** a GigabitEthernet interface.
Sub-command headings are displayed in ***red*** text.	
Next Mode entries display the CLI prompt after a command is entered	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

DHCP Commands

The following commands configure the Dynamic Host Configuration Protocol (DHCP) on the XSR.

bootfile

This command sets the name of the default boot image for a DHCP client. Depending on the client configuration inheritance, the command should be used from the proper mode. If it is specified from multiple modes, an override mechanism chooses the innermost config value, with host being innermost, then client-class and pool being the most general.

Syntax

bootfile *filename*

filename	Specifies the name of the file that is used as a boot image.

Syntax of the “no” Form

Use the *no* form of this command to delete the boot image name:

no bootfile

Mode

Any of the following command modes are available:

DHCP pool configuration: **XSR(config-dhcp-pool)#**

DHCP host configuration: **XSR(config-dhcp-host)#**

DHCP client class configuration: **XSR(config-dhcp-class)#**

Example

The following example specifies *roboboot* as the name of the boot file:

XSR(config-dhcp-pool)#bootfile roboboot

client-class

This command specifies the name of a DHCP client class. The XSR aggregates DHCP clients which will share the same configured attributes. Adding a client class to different DHCP pools in not permitted. For example, you cannot add client class *marketing* to both *pool1* and *pool2*.

Note: Adding a client class to different DHCP pools in not permitted. For example, you cannot add client class *marketing* to both *pool1* and *pool2*.

Syntax

client-class *name*

name	Designation of the client class using standard ASCII characters.

Syntax of the “no” Form

Use the *no* form of this command to remove the client class:

no client-class *name*

Mode

Either of the following command modes are available:

DHCP pool configuration: **XSR(config-dhcp-pool)#**

DHCP host configuration: **XSR(config-dhcp-host)#**

When specified from DHCP pool configuration mode, the CLI acquires DHCP class configuration sub-mode: **XSR(config-dhcp-class)#**

When specified from DHCP host configuration mode, the CLI does not acquire a new sub-mode.

Example

The following example specifies string *clientclass1* that will be the name of the client class:

```
XSR(config-dhcp-pool)#client-class cc1
```

client-identifier

This command specifies the unique identifier (in dotted hexadecimal notation) for a Microsoft DHCP client. It is valid for *manual* bindings only. Microsoft DHCP clients require client identifiers instead of hardware addresses. The client identifier is formed by concatenating the media type and the Ethernet hardware (MAC) address.

For example, the Microsoft client identifier for Ethernet address *0001.f401.2710* is *0100.01f4.0127.10*, where the leading *01* (italicized above) indicates the Ethernet media type. Be aware that you cannot add a client identifier to different DHCP pools. For example, client ID *0100.01f4.0127.10* cannot be added to both *pool1* and *pool2*.

Note: You cannot add a client identifier to different DHCP pools. For example, client ID *0100.01f4.0127.10* cannot be added to both *pool1* and *pool2*.

Syntax

client-identifier *identifier* [**client-class** *name*]

identifier	Unique identification of the client in dotted hexadecimal notation; for example: 0100.01f4.0127.10.
name	Specifies a client belonging to a client class.

Syntax of the “no” Form

Use the *no* form of this command to delete the client identifier:

no client-identifier *identifier* [**client-class** *name*]

Mode

Any of the following command modes are available:

DHCP pool configuration: **XSR(config-dhcp-pool)#**

DHCP host configuration: **XSR(config-dhcp-host)#**

DHCP client class configuration: **XSR(config-dhcp-class)#**

Next Mode

When this command is specified from DHCP pool configuration sub-mode or DHCP client-class mode, the CLI acquires DHCP host mode. When the command is entered from DHCP host mode, the CLI does not acquire a sub-mode.

```
XSR(config-dhcp-host)#
```

Example

The following example specifies the client identifier for MAC address 00.01f4.0127.10 in dotted hexadecimal notation:

```
XSR(config-dhcp)#client-identifier 0100.01f4.0127.10
```

The following example specifies the client identifier for MAC address *0001.f401.2710* in dotted hexadecimal notation, for the host with IP address *10.10.10.20*:

```
XSR(config-dhcp-pool)#host 10.10.10.20 255.255.255.0
XSR(config-dhcp-host)#client-identifier 0100.01f4.0127.10
```

The following example specifies the client identifier for MAC address *00.01f4.0127.10* in dotted hexadecimal notation, and adds it to class *eng*:

```
XSR(config-dhcp-pool)#client-class eng
XSR(config-dhcp-class)#client-identifier 0100.01f4.0127.10
```

client-name

This command specifies the name of a DHCP client. The client name should not include the domain name. The command is available from DHCP host mode *only*.

Syntax

client-name *name*

name	Designation of the client, defined using any set of standard ASCII characters. The client name should not include the domain name. For example, the name *soho* should not be specified as *soho.enterasys.com*.

Syntax of the “no” Form

Use the *no* form of this command to remove the client name:

no client-name *name*

Mode

DHCP host configuration *only*: **XSR(config-dhcp-host)#**

Example

The following example specifies a string *soho1* that will be the name of the client with MAC address *1111.2222.3333*:

```
XSR(config-dhcp-pool)#hardware-address 1111.2222.3333
XSR(config-dhcp-host)#client-name soho1
```

debug ip dhcp server

This command enables DHCP server debugging. This command should be used for troubleshooting purposes only.

Syntax

debug ip dhcp server {*events* | *packets* | *linkages*}

events	Reports server events, such as address assignments and database updates.
packets	Decodes DHCP receptions and transmissions.
linkages	Displays database linkage data such as parent-child relationships in a radix tree.

Syntax of the “no” Form

Use *no* form of this command to disable DHCP server debugging:

no debug ip DHCP server {*events* | *packets*}

Default

Disabled

Mode

Privileged EXEC: **XSR#**

Example

The following example enables DHCP server events debugging:

```
XSR#debug ip DHCP server events
```

default-router

This command specifies the default router list for a DHCP client. Depending on the client configuration inheritance, the command should be used from the proper mode. If it is specified from multiple modes, an override mechanism chooses the innermost config value, with *host* as innermost, then *client-class* and *pool* as the most general.

Syntax

default-router address [*address2*...*address8*]

address	IP address of a default router. One IP address is required.
address2 ...*address8*	Specifies up to eight addresses in the command line listed in order of preference (default router address has the highest priority, then router address 2, etc.).

Syntax of the “no” Form

Use the *no* form of this command to remove the default router list:

no default-router

Mode

Any of the following command modes are available:

DHCP pool configuration: **XSR(config-dhcp-pool)#**

DHCP host configuration: **XSR(config-dhcp-host)#**

DHCP client class configuration: **XSR(config-dhcp-class)#**

Example

The following example sets *14.12.1.99* as the IP address of the default router for any client in the subnet with three other routers in descending order of preference:

```
XSR(config-dhcp-pool)#default-router 14.12.1.99 14.13.1.66 14.12.1.56 14.12.1.57
```

The following example specifies *14.12.1.1* as the IP address of the default router for the host with MAC address *0010.a4f5.28a1*:

```
XSR(config-dhcp-pool)#hardware-address 0010.a4f5.28a1
XSR(config-dhcp-host)#default-router 14.12.1.1
```

The following example specifies *14.12.1.99* as the IP address of the default router for any client in the client class *eng*:

```
XSR(config-dhcp-pool)#client-class eng
XSR(config-dhcp-class)#default-router 14.12.1.99
```

dns-server

This command specifies the DNS IP servers available to a DHCP client. It is available from DHCP pool, host, or client class mode. Depending on the client configuration inheritance, the command should be used from the proper mode. If it is specified from multiple modes, an override mechanism chooses the innermost config value, with *host* as innermost, then *client-class* and *pool* as the most general.

Syntax

dns-server *address* [*address2...address8*]

address	IP address of a DNS server. One IP address is required.
address2 ... address8	You can list up to 8 addresses at the prompt line by order of preference (DNS server address is highest priority, then server address2, etc.).

Syntax of the “no” Form

Use the *no* form of this command to remove the DNS server list:

```
no dns-server
```

Mode

Any of the following command modes are available:

DHCP pool configuration: **XSR(config-dhcp-pool)#**

DHCP host configuration: **XSR(config-dhcp-host)#**

DHCP client class configuration: **XSR(config-dhcp-class)#**

Example

The following example specifies *11.12.1.99* as the IP address of the DNS server of a client in the subnet:

```
XSR(config-dhcp-pool)#dns-server 11.12.1.99
```

The following example specifies *11.12.1.99* as the IP address of the DNS server of the host with the MAC address *1111.2222.3333*:

```
XSR(config-dhcp-pool)#hardware-address 1111.2222.3333
XSR(config-dhcp-host)#dns-server 11.12.1.99
```

The following example specifies *11.12.1.99* as the IP address of the DNS server of a client in the client-class *engineering*:

```
XSR(config-dhcp-pool)#client-class engineering
XSR(config-dhcp-class)#dns-server 11.12.1.99
```

domain-name

This command specifies the domain name for DHCP client services by the DHCP server. Depending on the client configuration inheritance, the command should be used from the proper mode. If it is specified from multiple modes, an override mechanism chooses the innermost config value, with *host* as innermost, then *client-class* and *pool* as the most general.

Syntax

domain-name *domain*

domain	Domain name string of the client.

Syntax of the “no” Form

Use the *no* form of this command to remove the domain name:

```
no domain-name
```

Mode

Any of the following command modes are available:

DHCP pool configuration: **XSR(config-dhcp-pool)#**

DHCP host configuration: **XSR(config-dhcp-host)#**

DHCP client class configuration: **XSR(config-dhcp-class)#**

Examples

The following example specifies *enterasys.com* as the domain name of a client in the subnet:

```
XSR(config-dhcp-pool)#domain-name enterasys.com
```

The following example specifies *enterasys.com* as the domain name of the host with the MAC address *0011.a121.1fa2*:

```
XSR(config-dhcp-pool)#hardware-address 0011.a121.1fa2
XSR(config-dhcp-host)#domain-name enterasys.com
```

The following example specifies *enterasys.com* as the domain name of any client in the client-class *engineering*:

```
XSR(config-dhcp-pool)#client-class engineering
XSR(config-dhcp-class)#domain-name enterasys.com
```

hardware-address

This command sets the hardware address of a DHCP client and is valid for *manual* bindings *only.*

Note: You cannot add a hardware address to different DHCP pools. Hardware address *0100.01f4.0127.10* cannot be added to both *pool1* and *pool2*, e.g.

Syntax

hardware-address *address type* [**client-class** *name*]

address	MAC address of the client hardware platform.
type	Protocol of the hardware platform. Strings and values are acceptable. String options are: • ethernet • ieee802 Value options: • *1* - 10 Mbyte Ethernet • *6* - IEEE 802 networks
name	A client belonging to a client class can be specified here.

Syntax of the “no” Form

Use the *no* form of this command to remove the hardware address:

no hardware-address *address type* [**client-class** *name*]

Default

Ethernet

Mode

Any of the following command modes are available:

DHCP pool configuration: **XSR(config-dhcp-pool)#**

DHCP host configuration: **XSR(config-dhcp-host)#**

DHCP client class configuration: **XSR(config-dhcp-class)#**

Next Mode

When this command is entered from DHCP pool configuration sub-mode or DHCP client-class mode, the CLI acquires DHCP host configuration mode:

XSR(config-dhcp-host)#

When specified from either DHCP host or client mode, the command does not cause the CLI to acquire any sub-mode.

Examples

The following example specifies the hardware address for the DHCP client host to be of Ethernet type with MAC address *0001.f401.2710*:

```
XSR(config-dhcp-pool)#hardware-address 0001.f401.2710 ethernet
```

The following example specifies the hardware address for the DHCP client host with IP address *10.10.10.20* to be of Ethernet type with *0001.f401.2710* as the MAC address:

```
XSR(config-dhcp-pool)#host 10.10.10.20 255.255.255.0
XSR(config-dhcp-host)#hardware-address 0001.f401.2710 ethernet
```

The following example sets the hardware address for the DHCP host in class *eng* to be of Ethernet type with MAC address *0001.f401.2710*:

```
XSR(config-dhcp-pool)#client-class writer
XSR(config-dhcp-class)#hardware-address 0001.f401.2710 ethernet
```

host

This command specifies the IP address and network mask for a *manual* binding to a DHCP client. By default, the DHCP server will examine its defined IP address pools if the mask and prefix length are unspecified. If no mask is specified in the IP address pool database, the Class A, B, or C natural mask is used. This command is valid for manual bindings *only*.

Note: You cannot add a host to different DHCP pools. For example, host *firewall* cannot be added to both *pool1* and *pool2*.

Syntax

host *address* [*mask* | *prefix-length*]

address	IP address of the client.
mask	Network mask of the client.
prefix-length	Number of bits that comprise the address prefix. The prefix is an alternative way of specifying a client's network mask. It must be preceded by a forward slash (/).

Syntax of the "no" Form

Use the *no* form of this command to remove the IP address of the client:

```
no host
```

Mode

Any of the following command modes are available:

DHCP pool configuration: **XSR(config-dhcp-pool)#**

DHCP host configuration: **XSR(config-dhcp-host)#**

DHCP client class configuration: **XSR(config-dhcp-class)#**

Next Mode

When this command is specified from either DHCP pool configuration mode or DHCP class configuration sub-mode, the CLI acquires DHCP host configuration mode. When specified from DHCP host or client mode, the command does not acquire a sub-mode.

```
XSR(config-dhcp-host)#
```

Examples

This example sets *15.12.1.99* as the IP address of the client and *255.255.248.0* as its subnet mask:

```
XSR(config-dhcp-pool)#host 15.12.1.99 255.255.248.0
```

The following example specifies *15.12.1.99* as the IP address and *255.255.248.0* as the subnet mask, for the host with hardware address *1111.2222.3333*:

```
XSR(config-dhcp-pool)#hardware-address 1111.2222.3333
XSR(config-dhcp-host)#host 15.12.1.99 255.255.248.0
```

The following example specifies *15.12.1.99* as the IP address and *255.255.248.0* as the subnet mask for the client in the client-class *eng*:

```
XSR(config-dhcp-pool)#client-class eng
XSR(config-dhcp-class)#host 15.12.1.99 255.255.248.0
```

ip address dhcp

This command configures an interface as a DHCP Client. An Ethernet interface can be configured to use DHCP Client to acquire an IP address as well as other configuration parameters. Bootfile download is not supported.

Note: When an interface address is configured to be DHCP *negotiated* the only legal version of the *no* command is entered as **`no ip address dhcp`**.

Syntax

`ip address dhcp` [**`client-id`** *`client-identifier`*][**`hostname`** *`string`*]

Parameters

`client-identifier`	This value corresponds to Option 61 passed within DHCP packets. A DHCP server uses this value to index its database of address bindings. The value is expected to be unique for all clients in an administrative domain. It is intended that this value be either a MAC address or the symbolic ID of a port with a MAC address (e.g. FastEthernet 1.)
`hostname` *`string`*	The string corresponds to Option 12. The name may or may not be qualified with the local domain name. RFC-1035 character set restrictions are enforced.

Syntax of the “no” Form

The *no* form of this command disables DHCP client:

```
no ip address dhcp
```

Default

DCHP Client is *not active* on an interface

Mode

Interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example enables DHCP Client:

```
XSR(config)#interface FastEthernet1
XSR(config-if<F1>)#ip address dhcp
```

ip dhcp ping packets

This command specifies the number of packets a DHCP server sends to an IP address as part of a ping operation. The DHCP server pings an IP address before assigning the address to a requesting client. If the ping is unanswered, the DHCP server assumes that the address is not in use and assigns the address to the requesting client. Setting the number argument to a value of 0 turns off the DHCP server ping operation completely.

Syntax

ip dhcp ping packets *number*

number	Sum of ping packets sent before assigning the address to a requesting client.

Syntax of the “no” Form

Use the *no* form of this command to prevent the server from pinging IP addresses:

```
no ip dhcp ping packets
```

Default

Two packets

Mode

Global configuration: **XSR(config)#**

Example

The following example specifies six ping attempts by the DHCP server toward an IP address before stopping any further ping attempts:

```
XSR(config)#ip dhcp ping packets 6
```

ip dhcp ping timeout

This command specifies how long a DHCP server waits for a ping reply from an IP address.

Syntax

ip dhcp ping timeout *milliseconds*

milliseconds	The interval the DHCP server waits for a ping reply before it stops trying to reach an IP address for client assignment. The peak timeout is 10 seconds.

Syntax of the "no" Form

Use the *no* form of this command to restore the ping timeout default:

no ip dhcp ping timeout

Default

500 milliseconds

Mode

Global configuration: **XSR(config)#**

Example

The following example specifies that the DHCP server will wait 900 milliseconds for a ping reply before considering the ping a failure:

```
XSR(config)#ip dhcp ping timeout 900
```

ip dhcp pool

This command configures a DHCP server IP address pool. The XSR supports adding 1000 network addresses per pool and one DHCP pool per network. Class B or higher subnet masks are supported.

Note: The DHCP pool name *must* match the name given the IP local pool.

Syntax

ip dhcp pool *name*

name	A character string or integer which *match* the name you designate for the IP local pool.

Syntax of the "no" Form

Use the *no* form of this command to remove the address pool:

no ip dhcp pool *name*

Default

DHCP address pools are not configured

Mode

Global configuration: **`XSR(config)#`**

Next Mode

DHCP pool configuration: **`XSR(config-dhcp-pool)#`**

Example

The following example adds IP local pool *sales* with specified subnetworks and defines *sales* as the name of the DHCP server IP address pool:

```
XSR(config)#ip local pool sales 192.168.57.0/24
XSR(config)#ip dhcp pool sales
XSR(config-dhcp-pool)#
```

ip dhcp server

This command enables the DHCP Server features on the XSR. By default, DHCP server services are disabled on all XSR interfaces, which means that the DHCP server will not respond to client requests received on any XSR ports. DHCP Server can be enabled on a *FastEthernet*/*GigabitEthernet* primary interface and VLAN sub-interface. Secondary interface assignment is not supported.

Note: If either DHCP/BOOTP Relay (using **`ip helper-address`**) or DHCP Server is enabled on one FastEthernet/GigabitEthernet port, you cannot also configure the other service on the second Fast/GigabitEthernet port. The XSR permits either one or the other service to operate, not both.

Syntax

```
ip dhcp server
```

`server`	Enables/disables a DHCP server on a FastEthernet/GigabitEthernet port.

Syntax of the “no” Form

Use the *no* form of this command to disable DHCP server features:

```
no ip dhcp
```

Default

Disabled

Mode

Interface configuration: **`XSR(config-if<xx>)#`**

Example

The following example enables DHCP server on FastEthernet port 1:

```
XSR(config)#interface fastethernet 1
XSR(config-if<F1>)#ip dhcp server
```

ip local pool

This command, when issued multiply, configures a *local pool* of IP addresses to be used for a DHCP Server pool range. Use it in conjunction with the *no* form of to create one or more local address pools from which IP addresses are assigned when a remote peer connects.

Note: For clients that use a statically defined IP address (do not use DHCP to obtain an IP address), you must exclude that address from the local pool.

The command acquires IP Local Pool mode and makes available the following sub-commands:

- **`exclude`** - Bars a range of IP addresses from the local pool. Refer to page 15-97 for the sub-command definition.
- **`exit`** - Quits IP Local Pool configuration mode. Refer to page 15-97 for the sub-command definition.

Syntax

`ip local pool` *`pool-name subnet-address subnet-mask`*

`pool-name`	Name of a particular local address pool.
`subnet-address`	Base address of an IP subnet used to allocate IP addresses.
`subnet-mask`	Subnet mask of that IP subnet. All subnet address bits matching zero bits in the mask must also be zero; that is, subnet and mask must be zero.

Syntax of the “no” Form

Use the *no* form of this command to delete an IP address from the pool:

`no ip local pool` *`pool-name`*

Default

No address pools are configured

Mode

Global configuration: **`XSR(config)#`**

Next Mode

IP Local Pool configuration: **`XSR(ip-local-pool)#`**

Examples

The following example creates a local IP address pool named *marketing*, which contains all IP addresses in the range 203.57.99.0 to 203.57.99.255:

```
XSR(config)#ip local pool marketing 203.57.99.0 255.255.255.0
```

exclude

This sub-command of **ip local pool** bars the use of a range of IP addresses from an earlier created IP pool.

Syntax

exclude {*ip address*}{*number*}

ip address	Starting address to be excluded from pool.
number	Number of addresses to exclude, ranging from 1 to 65535.

Syntax of the “no” Form

The *no* form exempts the specified IP address from being excluded from the pool:

exclude {*ip address*}{*number*}

Mode

Local IP Pool configuration: **XSR(ip-local-pool)#**

Examples

The following example excludes the ten IP addresses between *192.168.57.100* and *192.168.57.110 f*rom local pool *HQ*:

```
XSR(config)#ip local pool HQ 192.168.57.0 255.255.255.0
XSR(ip-local-pool)#exclude 192.168.57.100 10
```

The following example negates the exclusion of IP addresses *192.168.57.105* and *192.168.57.106* from the earlier excluded range of IP addresses in local pool *HQ*:

```
XSR(config)#ip local pool HQ
XSR(ip-local-pool)#no exclude 192.168.57.105 2
```

exit

This sub-command of **ip local pool** quits IP Local Pool configuration mode.

Syntax

exit

Mode

IP Local Pool configuration: **XSR(ip-local-pool)#**

lease

This command configures the duration of the lease for an IP address that a DHCP server assigns to a DHCP client. The lease time set is the system default value which overrides the non-specified default value (one day).

If the client requests a lease period exceeding the period configured on the server, the lease interval offered by the server will equal that of the value configured by this command. If the client does not request a particular lease period - typical client behavior - it is granted the configured default value. Manual bindings are not held accountable to this lease period.

Depending on the client configuration inheritance, the command should be used from the proper mode. If it is specified from multiple modes, an override mechanism chooses the innermost config value, with *client-class* as innermost, then *pool* as most general.

Syntax

lease {*days* [*hours*] [*minutes*] | **infinite**}

days	Duration of the lease in days.
hours	Number of hours in the lease. A *days* value must be supplied before you can configure an *hours* value.
minutes	Number of minutes in the lease. *Days* and *hours* values must be set before you can configure a *minutes* value.
infinite	Duration of the lease is unlimited.

Syntax of the “no” Form

Use the *no* form of this command to restore the default value:

no lease

Default

One day

Mode

Either of the following command modes are available:

DHCP pool configuration: **XSR(config-dhcp-pool)#**

DHCP client class configuration: **XSR(config-dhcp-class)#**

Example

The following example configures a one-day lease:

```
XSR(config-dhcp-pool)#lease 1
```

The following example configures a one-hour lease:

```
XSR(config-dhcp-pool)#lease 0 1
```

The following example configures a one-minute lease:

```
XSR(config-dhcp-pool)#lease 0 0 1
```

netbios-name-server

This command configures NetBIOS Windows Internet Naming Service (WINS) name servers that are available to Microsoft DHCP clients. Depending on the client configuration inheritance, the command should be used from the proper mode. If it is specified from multiple modes, an override mechanism chooses the innermost config value, with *host* as innermost, then *client-class* and *pool* as the most general.

Syntax

netbios-name-server address [*address2...address8***]**

address	IP address of a NetBIOS WINS server. One address is needed.
address2 .. address8	Specifies up to eight addresses in the command line listed in order of preference (NetBIOS name server address has the highest priority, then server address2, etc.

Syntax of the "no" Form

Use the *no* form of this command to remove the NetBIOS name server list:

no netbios-name-server

Mode

DHCP Pool, Host, or Client Class config mode: **XSR(config-dhcp-pool)#, XSR(config-dhcp-host)#** or **XSR(config-dhcp-class)#**

Example

The following example specifies the IP address of a NetBIOS name server available to a Microsoft DHCP client in the subnet:

```
XSR(config-dhcp-pool)#netbios-name-server 13.12.1.90
```

The following example specifies the IP address of a NetBIOS name server available to the Microsoft DHCP client with client identifier *1111.2222.3333.4444*:

```
XSR(config-dhcp-pool)#client-identifier 1111.2222.3333.4444
XSR(config-dhcp-host)#netbios-name-server 13.12.1.90
```

The following example specifies the IP address of a NetBIOS name server available to a Microsoft DHCP client in the client class *engineering*:

```
XSR(config-dhcp-pool)#client-class engineering
XSR(config-dhcp-class)# netbios-name-server 13.12.1.90
```

netbios-node-type

This command configures the NetBIOS node type for Microsoft DHCP clients. Depending on the client configuration inheritance, the command should be used in proper mode. If it is specified from multiple modes, an override mechanism chooses the innermost config value, with *host* as innermost, then *client-class* and *pool* as the most general.

Syntax

netbios-node-type *type*

type	Specifies the NetBIOS node type. Valid types are: • *b-node* - Broadcast • *p-node* - Peer-to-peer • *m-node* - Mixed • *h-node* - Hybrid (recommended)

Syntax of the “no” Form

Use the *no* form of this command to remove the NetBIOS node type:

no netbios-node-type

Mode

Any of the following command modes are available:

DHCP pool configuration: **XSR(config-dhcp-pool)#**

DHCP host configuration: **XSR(config-dhcp-host)#**

DHCP client class configuration: **XSR(config-dhcp-class)#**

Example

This example sets NetBIOS name server type as hybrid for a Microsoft DHCP client in the subnet:

```
XSR(config-dhcp)#netbios node-type h-node
```

The following example specifies the NetBIOS name server type as hybrid for the Microsoft DHCP client with MAC address *0010.a4f5.28a1*:

```
XSR(config-dhcp-pool)#hardware-address 0010.a4f5.28a1
XSR(config-dhcp-host)#netbios node-type h-node
```

The following example specifies the NetBIOS name server type as hybrid for a Microsoft DHCP client in the client class *engineering*:

```
XSR(config-dhcp-pool)#client-class engineering
XSR(config-dhcp-class)#netbios node-type h-node
```

next-server

This command specifies the server from which the initial boot file will be loaded. The server can be designated either by IP address or hostname.

Syntax

next-server server [*hostname* | *ip_address*]

hostname	Designation of the server by name.
ip_address	Designation of the server by IP address.

Syntax of the "no" Form

Use the *no* form of this command to remove the next-server:

no next-server server [*hostname* | *ip_address*]

Mode

Any of the following command modes are available:

DHCP pool configuration: **XSR(config-dhcp-pool)#**

DHCP host configuration: **XSR(config-dhcp-host)#**

DHCP client class configuration: **XSR(config-dhcp-class)#**

Example

The following example specifies the IP address of a next-server:

```
XSR(config-dhcp-pool)next-server 192.168.57.4
```

option

This command configures DHCP server options/extensions. DHCP Server provides a framework for passing configuration data to hosts on a TCP/IP network. Configuration values and other control data are carried in tagged data items stored in the options field of the DHCP message.

The data items are also called options or client extensions. The current set of XSR-supported DHCP options and BOOTP vendor extensions are described in Table 15-1 on page 102 and generally in RFC-2132. Default values are defined in RFC-1122.

Depending on the client configuration inheritance, the command should be used from the proper mode. If it is specified from multiple modes, an override mechanism chooses the innermost config value, with *host* as innermost, then *client-class* and *pool* as the most general.

Syntax

option *code* {**ascii** *string* | **hex** *string* | **ip** *address*}

code	DHCP option code.
ascii *string*	An ASCII character string. Strings containing space must be enclosed with quotes. The following options are set with an ASCII string: 12, 14, 15, 17, 18, 40, 47, and 64.

hex *string*	Dotted hexadecimal data. Each byte in hexadecimal character strings is two hex digits - each byte can be separated by a period, colon, or white space. The following options are set with a hex value: 2, 13, 19, 20, 22-27, 29-31, 34-39, 43, 46,58, 59.
ip *address*	Specifies an IP address. The following options are set with an IP address: 1, 3-11, 16, 21, 28, 32, 33, 41, 42, 44, 45, 48, 49, 65, 68-76, and 118.

Syntax of the “no” Form

Use the *no* form of this command to remove the options:

no option code [**instance** *number*]

Default

Default instance number: *0*

Mode

Any of the following command modes are available:

DHCP pool configuration: **XSR(config-dhcp-pool)#**

DHCP host configuration: **XSR(config-dhcp-host)#**

DHCP client class configuration: **XSR(config-dhcp-class)#**

Note: Option examples are shown following the table.

Table 15-1 XSR-Supported DHCP Options

#	Protocol Name	Category/ Type	Default	Description
0	Pad	-	-	Causes subsequent fields to align on word boundaries. Length: 1 octet
1	Subnet Mask	Basic/ Address Mask	See descrip-tion	Client's subnet mask (RFC-950). If both Subnet Mask and Router options are specified in a DHCP reply, the Subnet Mask option must be expressed first. Length: 4 octets Default: Subnet of the interface on which the request was received
2	Time Offset	BOOTP/ 32-bit hex integer (in twos)	-	Offset of a client's subnet in seconds from Coordinated Universal Time (UTC). Positives indicate a site *east* of, and negatives a site *west* of the zero meridian. Length: 4 octets
3*	Router	Basic, MS DHCP Client/ IP address list	-	List of IP addresses for default routers on the client's subnet. List in order of preference. Length: 4-octet minimum; multiples of 4 CLI command: **default-router**
4	Time Server	BOOTP/IP address list	-	RFC-868 compliant timeservers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4

Table 15-1 XSR-Supported DHCP Options (continued)

#	Protocol Name	Category/ Type	Default	Description
5	Name Server	BOOTP/IP address list	-	IEN 116 name servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4
6*	Domain Name Server	Basic, MS DHCP Client/ IP address list	-	List of Domain Name System (STD 13, RFC-1035) name servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4 CLI command: **`dns-server`**
7	Log Server	Servers/IP address list	-	MIT-LCS UDP log servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4
8	Cookie Server	BOOTP/IP address list	-	RFC-865 compliant cookie servers available to the client. List in order of preference. Length: 4-octet minimum; multiples of 4
9	LPR Server	Servers/IP address list	-	RFC-1179 compliant line printer servers available to the client. List in order of preference. Length: 4-octet minimum; multiples of 4
10	Impress Server	BOOTP/IP address list	-	Imagen Impress servers available to the client. List in order of preference. Length: 4-octet minimum; multiples of 4
11	Resource Location Server	BOOTP/IP address list	-	RFC-887 compliant resource location servers available to the client. List in order of preference. Length: 4-octet minimum; multiples of 4
12*	Host Name	Basic/ASCII string	-	Name of the client which will or will not be qualified with the local domain name. See RFC-1035 for character set limits. Length: 1-octet minimum; multiples of 4 CLI command: **`client-name`**
13	Boot File Size	BOOTP/16-bit hex integer	-	Length in 512-octet blocks of the default boot image for the client. Length: 2 octets
14	Merit Dump File	BOOTP/ ASCII string	-	Path name of a file to which the client's core image will be placed if the client crashes. Use forward-slashes. Length: 4-octet minimum
15*	Domain Name	Basic, MS DHCP Client/ ASCII string	-	Domain name that the client will use when resolving host names through the Domain Name System. Length: 4-octet minimum CLI command: **`domain-name`**
16	Swap Server	BOOTP/IP address list	-	IP address of the client's swap server. Length: 4-octet minimum; multiples of 4
17	Root Path	BOOTP/ ASCII string	-	Path name of a client's root disk. Use forward-slashes. Length: 4-octet minimum
18	Extensions Path	BOOTP/ ASCII string	-	String specifying a file, retrievable through TFTP. Use forward-slashes. Length: 4-octet minimum
19	IP Forward-ing Enable /Disable	Host IP/ Boolean (hex)	false	Specifies if a client will set its IP layer for packet forwarding. Length: 1 octet Values: 0=disable; 1=enable

Table 15-1 XSR-Supported DHCP Options (continued)

#	Protocol Name	Category/ Type	Default	Description
20	Non-Local Source Routing	Host IP/ Boolean (hex)	false	Specifies whether a client will configure its IP layer to allow forwarding of datagrams with non-local source routes. Length: 1 octet Values: 0=disable; 1=enable
21	Policy Filter	Host IP/ Alternating IP address/ mask	-	Policy filters for non-local source routing, consisting of a list of IP addresses and masks that specify destination/mask pairs with which to filter incoming source routes. Any source-routed datagram whose next-hop address does not match one of the filters should be discarded by the client. Length: 8-octet minimum; multiples of 8
22	Maximum Datagram Reassem-bly Size	Host IP/16-bit hex integer	576	Peak size datagram a client will be ready to reassemble. Length: 2 octets Value: 576 minimum
23	Default IP Time-to-Live	Host IP/1 to 255 (hex), rejects 0	64	Default TTL that a client will use on outgoing datagrams. Length: 1 octet Values: 1 to 255
24	Path MTU Aging Timeout	Host IP/32-bit hex integer	-	Timeout (in seconds) to use when aging Path MTU values discovered by the mechanism (RFC-1191). Length: 4-octets
25	Path MTU Plateau Table	Host IP/16-bit hex integer	-	Table of MTU sizes to use when performing Path MTU Discovery (RFC-1191). It is ordered from smallest to largest. Length: 2-octet minimum, multiples of 2 Value: 68 minimum
26	Interface MTU	Interface/ 16-bit hex integer(s)	576	Maximum time to live on this interface. Length: 2-octet minimum; multiples of 2 Value: 68 minimum
27	All Subnets Are Local	Interface/ Boolean (hex)	false	Specifies if a client will assume all subnets of the IP network to which the client is connected use the same MTU as the subnet of that network to which the client is directly linked. Length: 1 octet Values: 1=all subnets share same MTU; 0=some directly-connected subnets may have smaller MTUs
28	Broadcast Address	Interface/ 0.0.0.0, 255.255.255. 255, or nonstandard	255.255. 255.255	Broadcast address in use on the client's subnet. Length: 4 octets
29	Perform Mask Discovery	Interface/ Boolean (hex)	false	Specifies if a client will perform subnet mask discovery via ICMP. Length: 1 octet Values: 0=disable; 1=enable
30	Mask Supplier	Interface/ Boolean (hex)	false	Specifies ifa client will respond to subnet mask requests via ICMP. Length: 1 octet Values: 0=do not respond; 1=respond
31	Perform Router Discovery	Interface/ Boolean	-	Specifies if a client will solicit routers using Router Discovery mechanism (RFC-1256). Length: 1 octet Values: 0=disable; 1=enable

Table 15-1 XSR-Supported DHCP Options (continued)

#	Protocol Name	Category/ Type	Default	Description
32	Router Solicitation Address	Interface/ IP address	-	Address to which a client should send router solicitation requests. Length: 4 octets
33	Static Route	Interface/ IP address pairs	-	Static routes that a client will install in its routing cache. If multiple routes to the same destination are specified, they are listed in descending order of priority. Routes consist of a list of IP address pairs: the first is the destination address, the second is the router for the destination. The default route 0.0.0.0 is an illegal destination for a static route. Length: 8-octet minimum; multiples of 8
34	Trailer Encapsu-lation	Interface/ Boolean (hex)	false	Specifies if a client will negotiate the use of trailers (RFC-893) when using the ARP protocol. Length: 1 octet Values: 0 = do not use; 1 = use
35	ARP Cache Timeout	Interface/ 32-bit hex integer	60	Timeout in seconds for ARP cache entries. Length: 4-octets
36	Ethernet Encapsu-lation	Interface/ Boolean (hex)	false (1.e., 894 style)	Specifies if a client will use Ethernet Version 2 (RFC-894) or IEEE 802.3 (RFC-1042) encapsulation if port is Ethernet. Length: 1 octet Value: 0 uses RFC-894 coding; 1 uses RFC-1042 coding
37	TCP Default TTL	Interface/ 8-bit integer (> 0)	60	Default TTL a client will use when sending TCP segments. Length: 1 octet, expressed in hex Value: minimum 1
38	TCP Keepalive Interval	Interface/ 32-bit hex integer	0 (keep--alives not generat-ed)	Interval in seconds that the TCP client will wait before sending a keep-alive message on a TCP connection. The time is specified as a 32-bit unsigned integer. A value of zero indicates that the client will not generate keep-alive messages on connections unless specifically requested by an application. Length: 4-octets
39	TCP Keepalive Garbage	Interface/ Boolean (hex)	false (off)	Specifies if a client will send TCP keep-alive messages with an octet of garbage for compatibility with older implementations. Length: 1 octet Values: 0=do not send; 1=send
40	NIS Domain	Servers/ ASCII string	-	Name of a client's NIS domain. Length: 4-octet minimum
41	Network Information Servers	Servers /IP address list	-	IP addresses indicating NIS servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4
42	NTP Servers	Servers /IP address list	-	IP addresses indicating NTP servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4
43	Vendor-Specific Data	- /Hex	-	Option used by clients/servers to swap vendor-specific data. Length: 4-octet minimum

Table 15-1 XSR-Supported DHCP Options (continued)

#	Protocol Name	Category/ Type	Default	Description
44*	NetBIOS over TCP/ IP Name Server	WINS/ NetBIOS, MS DHCP Client/ IP address list	-	RFC-1001/1002 NBNS name servers listed by preference. Length: 4-octet minimum; multiples of 4 CLI command: **`netbios-name-server`**
45	NetBIOS over TCP/ IP Datagram Distribution Server	WINS/ NetBIOS /IP address list	-	NBDD name servers(RFC-1001/1002) listed by preference. Length: 4-octet minimum; multiples of 4
46*	NetBIOS over TCP/ IP Node Type	WINS/ NetBIOS, MS DHCP Client/ 1, 2, 4, or 8 (hex)	-	The value is a single octet that identifies client type: 1: B-node; 2: P-node; 4: M-node; 8: H-node Length: 1 octet CLI command: **`netbios-node-type`**
47	NetBIOS over TCP/ IP Scope	WINS/ NetBIOS, MS DHCP Client/ ASCII string	-	NetBIOS over TCP/IP scope value for a client (RFC-1001/1002). Length: 4-octet minimum
48	X Windows Font Server	Servers/ IP address list	-	X Window System Font servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4
49	X Windows Display Manager	Servers/ IP address list	-	IP addresses of systems running X Window System Display Manager and are available to a client. List addresses in order of preference. Length: 4-octet minimum; multiples of 4
50	Requested IP Address	IP address	-	Used in a client request (DHCPDISCOVER or DHCPREQUEST) to allow a client to request a particular IP address be assigned. Length: 4 octets
51	IP Address Lease Time	Lease Information, MS DHCP Client/32-bit hex integer	-	Used in a client request (DHCPDISCOVER or DHCPREQUEST) to allow a client to request a lease time for the IP address. In a server reply (DHCPOFFER), a DHCP server uses this option to specify the lease time it is willing to offer. Length: 4 octets Value: seconds
52	Option Overload	-	-	Indicates that the DHCP *sname* or *file* fields are being overloaded by using them to carry DHCP options. A DHCP server inserts this option if the returned values will exceed the usual space allotted for options. If this option is present, the client interprets the specified additional fields after it concludes interpretation of the standard option fields. 1 = The file field is used to hold options. 2 = The sname field is used to hold options. 3 = Both fields are used to hold options. Length: 1 octet

Table 15-1 XSR-Supported DHCP Options (continued)

#	Protocol Name	Category/ Type	Default	Description
53	DHCP Message Type	-	-	Conveys the type of DHCP message. The default is 1 (DHCPDISCOVER). 1=DHCPDISCOVER 2=DHCPOFFER 3=DHCPREQUEST 4=DHCPDECLINE 5=DHCPACK 6=DHCPNAK 7=DHCPRELEASE 8=DHCPINFORM Length: 1 octet
54	Server Identifier	IP address	-	Used in DHCPOFFER and DHCPREQUEST messages, and may optionally be included in the DHCPACK and DHCPNAK messages. DHCP servers include this option in the DHCPOFFER to allow the client to distinguish between lease offers. DHCP clients use the contents of the server identifier field as the destination address for any DHCP messages unicast to the DHCP server. DHCP clients also indicate which of several lease offers is being accepted by including this option in a DHCPREQUEST message. The identifier is the IP address of the selected server. Length: 4 octets
55	Parameter Request List	Hex integer	-	Used by a DHCP server to request values for specified configuration parameters. The list of requested values is specified as *n* octets, where each octet is a valid DHCP option code. The client can list the options in order of preference. The DHCP server is not required to return the options in the requested order, but must try to insert the requested options in the order requested by the client. Length: 1-octet minimum
56	Message	String	-	Used by a DHCP server to print an error message to a DHCP client in a DHCPNAK message in the event of a failure. A client may use this option in a DHCPDECLINE message to indicate why the client declined the offered values. The message consists of *n* octets of NVT ASCII text, which the client may display on an available output device. Length: 1-octet minimum
57	Maximum DHCP Message Size	16-bit hex integer	-	Maximum length DHCP message that a client is willing to accept. Length is specified as an unsigned 16-bit integer. A client may use the maximum DHCP message size option in DHCPDISCOVER or DHCPREQUEST messages, but should not use the option in DHCPDECLINE messages. Length: 2 octets Value: 576 minimum
58	Renewing (T1) Time Value	Lease Data, MS DHCP Client/32-bit hex integer	-	Time interval from address assignment until a client transitions to the RENEWING state. Length: 4 octets Value: seconds, as a 32-bit unsigned integer
59	Rebinding (T2) Time Value	Lease Data, MS DHCP Client/32-bit hex integer	-	Interval from address assignment until a client transitions to the REBINDING state. Length: 4 octets Value: seconds, as a 32-bit unsigned integer

Table 15-1 XSR-Supported DHCP Options (continued)

#	Protocol Name	Category/ Type	Default	Description
61 *	Client-Identifier	Basic/String	-	A DHCP client's unique identifier. DHCP servers use this value to index their database of address bindings. This value is expected to be unique for all clients in an administrative domain. Length: 2-octet minimum CLI command: **`ip address dhcp`**
64	NIS+ Domain	Servers/ ASCII string	-	Name of the client's NIS+ domain. Length: 4-octet minimum
65	NIS+ Servers	Servers/IP address list	-	IP addresses indicating NIS+ servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4
67	Bootfile name	BOOTP/ String	-	Identifies a bootfile name when when the *file* field in the DHCP header has been used for DHCP options. Length: 1-octet minimum
68	Mobile IP Home Agent	Servers/IP address list	-	IP addresses indicating mobile IP home agents available to a client. List agents in order of preference. Length: 4-octet minimum; multiples of 4
69	SMTP Server	Servers/IP address list	-	SMTP servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4
70	POP3 Server	Servers/IP address list	-	POP3 servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4
71	NNTP Server	Servers/IP address list	-	NNTP servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4
72	Default WWW Server	Servers/IP address list	-	WWW servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4
73	Default Finger Server	Servers/IP address list	-	Finger servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4
74	Default IRC Server	Servers/IP address list	-	IRC servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4
75	StreetTalk Server	Servers/IP address list	-	StreetTalk servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4
76	STDA Server	Servers/IP address list	-	STDA servers available to a client. List in order of preference. Length: 4-octet minimum; multiples of 4
82	DHCP Relay Agent Information	DHCP Relay/ String	-	This *helper* option is used in an environment where DHCP Relay is co-located with circuit access equipment (DSL and cable-based LANs) to reduce broadcasts, prevent IP spoofing, client ID spoofing, and MAC address spoofing. Defined by RFC-3046. Length: Variable
90	DHCP Authentica-tion	DHCP Protocol/ Structured Data	-	Mechanism for authenticating DHCP messages, clients and servers. Based on HMAC-MD5. Defined by RFC-3118. Length: Variable; minimum 11 octets

Table 15-1 XSR-Supported DHCP Options (continued)

#	Protocol Name	Category/ Type	Default	Description
117	Name Service Search	Server/ Multiple 16-bit hex integers	-	Sets site of Name Service servers to clients to be used for lookup. Each 16-bit field specifies a Name Server to be used for lookup: 0 – client should refer to local naming information 6 – use DNS 41 – use NIS 44 – use NetBIOS over TCP/IP 65 – use NIS+ Defined by RFC-2937. Length: Minimum 2 octets; multiple of 2 octets
118	Subnet Selection	Interface/ IP address	-	Sets the subnet IP address (RFC-3011). Used by a client to inform/force server to assign an IP address-specific subnet. Length: 4 octets
150	TFTP Server	Cisco Vendor Extension/IP address	-	Address of the TFTP server. This option supports the XSR's Remote Auto Install functionality. Length: 4 octets

Note: DHCP options marked with an asterisk (*) can also be configured at the CLI.

Examples

The following example configures DHCP option 33, which specifies static routes that the client should install in its routing cache. If multiple routes to the same destination are set, they are listed in descending order of priority. The routes consist of IP address pairs. The first address is the destination address, the second address is the router for the destination.

```
XSR(config-dhcp-pool)#option 33 ip 90.1.1.90 123.124.23.26 90.1.1.90 123.24.56.78
```

The following example configures DHCP option 19, which specifies whether the client should enable its IP layer for packet forwarding. Values of *0* and *1* disable and enable IP forwarding, respectively. IP forwarding is enabled in the following example:

```
XSR(config-dhcp-pool)#option 19 hex 01
```

The following example configures DHCP option 1, which sets the client's subnet mask as higher priority when it and the router ID are specified in the DHCP REPLY:

```
XSR(config-dhcp-pool)#option 1 ip 255.255.255.0
```

The following example configures DHCP option 2, which locates a client as an offset 4650 seconds from Coordinated Universal Time (UTC) or five hours west of the zero meridian (London):

```
XSR(config-dhcp-pool)#option 2 hex 4650
```

The following example configures DHCP option 72, which specifies World Wide Web (WWW) servers for DHCP clients. Two WWW server addresses are configured in the following example:

```
XSR(config-dhcp-pool)#option 72 ip 168.24.3.252 168.24.3.253
```

The example below configures DHCP option 13, which specifies a client's default boot image size:

```
XSR(config-dhcp-pool)#option 13 hex 8001
```

The following example configures DHCP option 41, which specifies Network Information Servers (NIS) for DHCP clients. Two NIS server addresses are configured in the following example:

```
XSR(config-dhcp-pool)#option 41 ip 90.3.4.5 90.1.1.7 90.43.9.254
```

The following example configures DHCP option 36, which specifies Ethernet encapsulation Version 2 (RFC-894) or IEEE 802.3 for DHCP clients. Version 2 encapsulation is set in this example:

```
XSR(config-dhcp-pool)#option 36 hex 00
```

The following example configures DHCP option 21, which sets a policy filter for non-local source routing. The filters consist of a list of IP addresses and masks that specify destination/mask pairs with which to filter inbound source routes. Any source-routed datagram whose next-hop address does not match one of the filters is discarded by the client.

```
XSR(config-dhcp-pool)#option 21 ip 90.1.1.78 255.255.0.0 134.141.90.1
255.255.255.0
```

The following example configures DHCP option 22, which specifies the maximum size datagram a client will reassemble. The value is 1052 bytes:

```
XSR(config-dhcp-pool)#option 22 hex 41
```

The following example sets DHCP option 28, specifying the broadcast address in use on the client's subnet. The value is: 255.255.255.255.

```
XSR(config-dhcp-pool)#option 28 ip 255.255.255.255
```

The following example configures DHCP option 35, which specifies the timeout in seconds for ARP cache entries. The value is 604,800 (1 week):

```
XSR(config-dhcp-pool)#option 35 hex 93A8
```

The following example sets DHCP option 14, specifying the pathname where a DHCP client's core image will be placed if the client crashes:

```
XSR(config-dhcp-pool)#option 14 ascii c:/dump/path
```

The following example configures DHCP option 31, which specifies that the DHCP client should not perform subnet mask discovery:

```
XSR(config-dhcp-pool)#option 29 hex 00
```

The following example configures DHCP option 19, which specifies that the DHCP client should configure its IP layer for packet forwarding:

```
XSR(config-dhcp-pool)#option 19 hex 01
```

The following example configures DHCP option 31, which specifies that the DHCP client should perform Router Discovery:

```
XSR(config-dhcp-pool)#option 31 hex 01
```

The following example configures DHCP option 47, which specifies a NetBIOS over TCP/IP scope parameter for a DHCP client:

```
XSR(config-dhcp-pool)#option 47 ascii scope
```

The following example configures DHCP option 40, which specifies the DHCP client's NIS domain:

```
XSR(config-dhcp-pool)#option 40 ascii NISserver
```

The following example configures DHCP option 18, which specifies the pathname of a file retrievable through TFTP:

```
XSR(config-dhcp-pool)#option 18 ascii /extension/path
```

The following example configures DHCP option 18, which specifies a list of prioritized static routes (in descending order) the DHCP client should install in its routing cache:

```
XSR(config-dhcp-pool)#option 33 ip 90.1.1.90 123.124.23.26 90.1.1.90 123.24.56.78
```

service dhcp

This command enables DHCP server functionality to respond to client requests. Although DHCP server is enabled by default on all XSR interfaces, you can optionally enable or disable it on a specific interface.

Syntax

service dhcp [*interface*]

interface	The port on which the DHCP server is enabled or disabled.

Syntax of the "no" Form

Disable the DHCP server by using the *no* form of this command:

no service dhcp [*interface*]

Default

Enabled on all interfaces

Mode

Global configuration: XSR(config)#

Example:

The example below enables DHCP services on interface FastEthernet 1:

```
XSR(config)#service dhcp fastethernet 1
```

DHCP Clear and Show Commands

clear ip dhcp binding

This command deletes an automatic address binding from the DHCP server binding database. Use the **no host** command to delete a *manual* binding. Typically, the address denotes the IP address of the client. If an asterisk (*) is used as the address parameter, DHCP clears all automatic bindings.

Syntax

clear ip dhcp binding {*address* | * }

address	Address of the binding you want to clear.
*	Clears all automatic bindings.

Mode

Privileged EXEC: **XSR#**

Example

The example below deletes address binding *18.12.22.99* from a DHCP server bindings database:

```
XSR#clear ip dhcp binding 18.12.22.99
```

clear ip dhcp server statistics

This command resets all DHCP server counters. All counters are cumulative and are initialized, or set to zero, with this command.

Syntax

```
clear ip DHCP server statistics
```

Mode

Privileged EXEC: **XSR#**

Example

The following example resets all DHCP counters to zero:

```
XSR#clear ip DHCP server statistics
```

show dhcp lease

This command displays DHCP Client information.

Syntax

```
show dhcp lease
```

Mode

Privileged EXEC: **XSR#**

Example

```
XSR#show dhcp lease
Temp IP addr: 192.168.70.102 for peer on Interface: FastEthernet0
Temp sub net mask: 255.255.255.0
Temp default-gateway addr: 192.168.70.1
State: 5 BOUND
   DHCP Lease Server: 192.168.70.1, config.enterasys.com
   DNS Server: 24.25.26.27
               24.25.26.28
   DHCP transaction id: 29247
   Lease: 36000 secs, Renewal: 17205 secs, Rebind: 31500 secs
   Next timer fires after 4:44:25
```

Parameter Descriptions

Temp IP addr	IP address assigned via DHCP to the client from the server.
Temp sub net mask	Subnet mask assigned via DHCP to the client from the server.
Temp default-gateway addr	Default gateway assigned by the DHCP server.
State	DHCP Client FSM state: • 0 - None • 1 - REBOOTING • 2 - INIT • 3 - SELECTING • 4 - REQUESTING • 5 - BOUND • 6 - RENEWING • 7 - REBINDING • 8 - STOPPED • 9 and others - NOTVALID
DHCP Lease Server	DHCP server IP address and name.
DNS Server	DNC server IP address.
DHCP Transaction ID	Transaction ID for current DHCP offer from the server.
Lease/ Renewal/ Rebind	Current lease, renewal, and rebind periods.
Next timer fires after	Timer for the next time DHCP renew request.

show interface

This command displays DHCP interface's IP address and subnet mask. When negotiating, the interface will indicate "Internet address is not assigned".

Syntax

```
show interface
```

Examples

The following example does not display the DHCP assigned address *while* the protocol is negotiating:

```
XSR#show interface
FastEthernet 1 is Admin Up
Internet address is not assigned
```

The following example displays the DHCP assigned address when the protocol *has finished* negotiation:

```
xsr#show interface
FastEthernet 1 is Admin Up
Internet address is 172.16.1.1, subnet mask is 255.255.255.0
```

show ip dhcp binding

This command displays active address bindings on the DHCP server. If the address is not specified, all address bindings are shown. Otherwise, only the binding for the specified client is displayed. The lease expiration time can be displayed based on the Universal Time Clock (UTC) or local clock. If the local clock is not specified, UTC is the default.

Note: BOOTP bindings do not have leases: their *Active* designation is always N.

Syntax

show ip dhcp binding [*ip-address*][*utc* | *local*]

ip-address	IP address of the DHCP client.
utc	Bindings displayed according to the Universal Time Clock.
local	Bindings displayed according to local time.

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Examples

The following examples display the lease expiration in default UTC time:

```
XSR#show ip dhcp binding 168.16.22.11

IP address   Hardware address    Lease expiration     Type       Act.
168.16.1.11  00a0.9802.32df      Feb 01 1998 12:00AM  Automatic  Y

XSR#show ip dhcp binding 168.16.22.254

IP address   Hardware address    Lease expiration      Type      ACT.
168.16.3.254 02c7.f800.0423      Infinite              Manual    N
```

The following example displays the lease expiration in local time:

```
XSR#show ip dhcp binding local

IP address     Hardware address    Lease expiration     Type     Act.
11.1.0.253     0002.2ab4.4b01      JUL 19 2003 09:07PM  Automatic  Y
```

The following example displays the lease expiration in UTC time:

```
XSR#show ip dhcp binding UTC

IP address     Hardware address     Lease expiration      Type       Act.
11.1.0.253     0002.2ab4.4b01      JUL 19 2003 05:07PM    Automatic  Y
```

The following example displaysthe lease expiration of DHCP client 11.1.0.253 in UTC time:

```
XSR#show ip dhcp binding UTC 11.1.0.253

IP address     Hardware address     Lease expiration      Type       Act.
```

```
11.1.0.253      0002.2ab4.4b01      JUL 19 2003 05:07PM    Automatic    Y
```

The following example the displays lease expiration of DHCP client 11.1.0.253 in local time:

```
XSR#show ip dhcp binding local 11.1.0.253

IP address      Hardware address      Lease expiration      Type        Act.
11.1.0.253      0002.2ab4.4b01        JUL 19 2003 09:07PM   Automatic    Y
```

Parameter Descriptions

IP address	IP address of the DHCP client.
Hardware address	Ethernet MAC address of the DHCP client.
Lease expiration	Date and time when the DHCP client's lease expires.
Type	*Automatic* or *Manual* lease renewal.
Act(tive)	Whether lease is active or not - *Y* or *N*.

show ip dhcp server statistics

This command displays DHCP server statistics.

Syntax

```
show ip dhcp server statistics
```

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Example

The following example displays DHCP server statistics:

```
XSR# show ip DHCP server statistics

Database agents       1

Memory usage          20392
Address pools         2
Database agents       1
Automatic bindings    26
Manual bindings       1
Expired bindings      3
Malformed messages    0

Message               Received
BOOTREQUEST           12
DHCPDISCOVER          20
DHCPREQUEST           17
DHCPDECLINE           0
DHCPRELEASE           0
DHCPINFORM            0
```

```
Message                 Sent
BOOTREPLY               12
DHCPOFFER               19
DHCPACK                 17
DHCPNAK                 6
```

Parameter Descriptions

Memory usage	Sum of bytes of RAM allocated by the DHCP server.
Address pools	Sum of configured address pools in the DHCP database.
Database agents	Sum of database agents entered in the DHCP database.
A*utomatic bindings*	Sum of IP addresses *automatically* mapped to the Ethernet MAC addresses of hosts found in the DHCP database.
Manual bindings	Sum of IP addresses *manually* mapped to the Ethernet MAC addresses of hosts found in the DHCP database.
Expired bindings	Sum of expired leases.
Malformed messages	Sum of truncated or corrupted messages received by the DHCP server.
Message	DHCP message type received by the DHCP server.

16

Configuring Security

Observing Syntax and Conventions

The CLI Syntax and conventions use the notation described in the following table.

Convention	Description
xyz	Key word or mandatory parameters (**bold**)
[*x*]	[ ] Square brackets indicate an optional parameter (*italic*)
[x | y | z]	[|] Square brackets with vertical bar indicate a choice of values
{x | y | z}	{ | } Braces with vertical bar indicate a choice of a required value
[x {y | z}]	[{ | }] Combination of square brackets with braces and vertical bars indicates a required choice of an optional parameter
(config-if<*xx*>)	*xx* signifies the interface type and number; e.g., `F1`, `G3`, `S2/1.0,M57`. `F` indicates a FastEthernet, and `G` a GigabitEthernet interface.
Next Mode entries display the CLI prompt after a command is entered.	
Sub-command headings are displayed in ***red***, italicized text.	
soho.enterasys.com	Italicized, non-syntactic text indicates either a user-specified entry or text with special emphasis

The following set of commands allows you to define security features for the XSR, including:

- "General Security Commands" on page 16-84
- "Security Clear and Show Commands" on page 16-91
- "AAA Commands" on page 16-93
- "AAA Usergroup Commands" on page 16-94
- "AAA User Commands" on page 16-97
- "AAA Method Commands" on page 16-101
- "AAA Per-Interface Commands" on page 16-111
- "AAA Debug and Show Commands" on page 16-112
- "Firewall Feature Set Commands" on page 16-115
- "Firewall Interface Commands" on page 16-129
- "Firewall Show Commands" on page 16-133

General Security Commands

access-list (extended)

This command defines an extended IP Access List (ACL) by number ranging from 100 to 199. You can restrict or allow the following traffic:

- IP (Any Internet Protocol)
- TCP (Transmission Protocol)
- UDP (User Datagram Protocol)
- ICMP (Internet Control Message Protocol)
- ESP (Encapsulation Security Payload)
- GRE (Generic Router Encapsulation) protocol
- AH (Authentication Header) protocol

New and existing ACL entries can be added/replaced in a particular ACL without you having to rewrite the entire ACL by using the *insert/replace number* parameters. If neither the *insert* nor the *replace* option is specified, then the new entry is appended to the list. This is noteworthy since ACL criteria are evaluated in the order displayed by the **show access-list** command.

Apply restrictions defined by an ACL with **ip access-group** command.

Syntax

```
access-list list# [insert | replace} entry# {deny | permit}{protocol}|{log}
{srcIpAddr [srcWildCardBits]| [qualifier] | source-port |
host srcIpAddr | any}        range min-sport | max-sport
{dstIpAddr [dstWildCardBits]| [qualifier]|destn-port |
host dstIpAddr | any}                                    [established]
                             range min-dprt | max-dprt
                             type [code]
```

list#	Extended ACL number, ranging from **100** - **199**.
insert	New access entry is inserted before existing *entry #* in the existing ACL. The **show access-list** command from within Global mode sequentially numbers entries for this purpose.
replace	New access entry replaces an entry # in the existing ACL (the entry # must already exist.)
entry#	Entry's list number within the ACL. No number is required for first entry.
deny	Access is denied if specified conditions are met.
permit	Access is permitted if conditions met.
protocol	Specifies the IP protocol: IP, TCP, UDP, ICMP, ESP, GRE, or AH. IP represents any protocol.
log	Enables alarm logging and reporting of source IP addresses for configured ACL entries.
srcIPAddr	The source expressed by IP address.

srcWild CardBits	Specifies bits to *ignore* in the source address. **Note:** The srcWildCardBits/dstWildCardBits mask specifies bits to *ignore* (which allow any value where the bits are set), as opposed to the traditional method of specifying bits to *keep*.
host	Only the exact source address matches the condition. Same as srcWildCardBits = 0.0.0.0.
any	Any source address matches the condition. Same as srcWildCardBits = 255.255.255.255.
qualifier	Value applied to the source port: **eq** - equal than, **neq** - not equal to, **lt** - less than, **gt** - greater than.
source-port	*Optional* source port number (0 - 65535).
range	Value must be within the minimum and maximum source and destination port range.
min-sport	Lowest port number from 0 to 65535. Combine with *max-sport*.
max-sport	Highest port number from 0 to 65535. Normally greater than *min-sport* but if less than *min*, values are swapped.
dstIPAddr	The destination expressed by IP address.
dstWild CardBits	Specifies bits to *ignore* in the destination address.
destn-port	Destination port number. Range: 0 to 65535.
type,*code*	ICMP message type *only* (0-255) and code (0-255).
established	Matches if a TCP connection is already established, that is, if either ACK or RST bits are set in the TCP header.

Note: Source and destination ports are defined only for TCP or UDP. A message type and code can be defined for ICMP.

Additional Syntax

The **access-list** command also provides the **move** option, expressed in the following syntax:

access-list *list-number* **move** *destination src1* [*src2*]

list#	ACL number, ranging from **100** - **199**.
move	Moves a sequence of ACL entries in front of another entry. Range: 1-999.
destination	Number of the existing ACL entry before which subsequent entry or range of entries is to be moved. Range: 1 to 999. If being moved to the end, use a non-existent number (e.g., 999).
src1	Single entry number, or the first entry number in the range to be moved before the destination. Range: 1 to 999.
src2	Optional last entry number in the range to be moved. Range: 1 to 999. If not specified, only one entry is moved.

Syntax of the “no” Form

The *no* form of this command removes the defined access list:

no access-list *list-number* [*ent1*][*ent2*]]

list#	The standard access list number, ranging from 1 to 99.
ent1	Optional single entry number, or the first entry number in the range to be removed. If unspecified, the entire ACL is removed.
ent2	Optional last entry number in the range to be removed.

Mode

Global configuration: **XSR(config)#**

Default

No access list defined (that is, all access permitted)

Examples

The following example denies access only for ICMP packets coming from hosts on the three specified networks. The wildcard bits apply to the host portions of the network addresses. Any host with a source address that does not match the access list statements will be permitted.

```
XSR(config)#access-list 100 deny ICMP 192.5.34.0 0.0.0.255
XSR(config)#access-list 100 deny ICMP 128.88.0.0 0.0.255.255
XSR(config)#access-list 100 deny ICMP 36.0.0.0 0.255.255.255
```

The following example replaces entry *87* with the following entry:

```
XSR(config)#access-list 123 replace 87 deny ip host 1.2.1.2
```

The following example removes entries *16*, *17* and *18* from ACL *177*:

```
XSR(config)#no access-list 177 16 18
```

The following example removes the entire ACL *102*:

```
XSR(config)#no access-list 102
```

The following example moves entries *16 - 18* within an ACL to the beginning of the list:

```
XSR(config)#access-list 101 move 1 16 18
```

The example below moves entries *16 - 18* from ACL *144* to its beginning:

```
XSR(config)#access-list 144 move 1 16 18
```

The following example moves entry *2* to the end of ACL *133*:

```
XSR(config)#access-list 133 move 999 2
```

access-list (standard)

This command defines a standard IP Access List (ACL) by numbers, ranging from 1 to 99. ACL restrictions are applied using the **ip access-group** command.

New and existing ACL entries can be added/replaced in a particular ACL without you having to rewrite the entire ACL by using the *insert/replace number* parameters. If neither the *insert* nor the *replace* option is specified, then the new entry is appended to the list. This is noteworthy since ACL criteria are evaluated in the order displayed by the **show access-list** command.

Syntax

access-list *list#* [[{**insert** | **replace** | **move**}] [{*entry# destination source1* [*source2*]]}{**deny** | **permit**}{log} {*srcIpAddr* [*srcWildCardBits*]| **host** *srcIpAddr* | **any**}

list#	Standard access list number ranging from 1 to 99.
insert	New access entry is inserted before an existing *entry #* in an ACL. The **show access-list** command sequentially numbers entries for this purpose.
replace	Same as above, except the new access entry replaces an *entry #* in the existing ACL (the *entry #* must already exist.)
move	Moves a sequence of ACL entries in front of another entry.
entry#	Sequential entry number in ACL to add/delete ranging from 1 to 999.
destination	Position before which entries are to be moved. Range: 1-999.
source1	Sequential number of first ACL entry to move. Range: 1-999.
source2	Sequential number of last ACL entry to move. Range: 1-999.
deny	Denies access if specified conditions are met.
permit	Permits access if conditions met.
log	Enables alarm logging and reporting of source IP addresses for configured ACL entries.
srcIpAddr	Identifies the source by IP address.
srcWildCard Bits	Bits to ignore in the source address. A mask of 0.0.0.225 implies only the most important bits of the source address are considered.
host	Marks only the exact source address matching the condition. Same as *srcWildCardBits* = 0.0.0.0.
any	Marks any source address matching the condition. Same as *srcWildCardBits* = 255.255.255.255.

Syntax of the “no” Form

The *no* form of this command removes the defined access list or entries (one or more) in a list:

no access-list *list-number* [*ent1* [*ent2*]]

list-number	The standard access list number ranging from 1 to 99.
ent1	Optional single entry number, or the first entry number in the range to be removed. If unspecified, the entire ACL is removed.
ent2	Optional last entry number in the range to be removed.

Mode

Global configuration: **XSR(config)#**

Default

No access list defined (all access permitted)

Examples

The following example allows access only to those hosts on the three specified networks. The wildcard bits apply to the host portions of the network addresses. Any host with a source address that does not match the access list statements will be rejected.

```
XSR(config)#access-list 1 permit 192.5.34.0 0.0.0.255
XSR(config)#access-list 1 permit 128.88.0.0 0.0.255.255
XSR(config)#access-list 1 permit 36.0.0.0 0.255.255.255
```

The following example replaces entry *88* with the following entry:

```
XSR(config)#access-list 57 replace 88 deny host 1.2.1.2
```

The example below removes entries *16*, *17* and *18* from ACL *87*:

```
XSR(config)#no access-list 87 16 18
```

The following example removes the entire ACL *57*:

```
XSR(config)#no access-list 57
```

The next example moves entries *16 - 18* from ACL *57* to its start:

```
XSR(config)#access-list 57 move 1 16 18
```

The example below moves entry *2* to the end of ACL *57*:

```
XSR(config)#access-list 57 move 999 2
```

access-list log-update-threshold

This command publishes an ACL violations log when a specified number of packets the XSR processes is met. ACL violations logging is updated every five minutes so regardless of how you specify this command, the five-minute timer remains in effect. The command functions as follows:

- ACL alarms display the: *ACL group number*, *permit* or *deny* clause, *source IP address* and *number of packets* logged in the last five minutes.
- Alarms are *set* to **medium** severity level by default.
- Setting the alarm severity level to **high** with the **`logging`** command *disables* all ACL alarms.
- After an update is reported, the log is *cleared* for the entry with that source IP and ACL group.
- *Standard* and *extended* ACLs are supported.
- If reporting is enabled for *every* packet, too many packets may log messages resulting in some message loss due to packet flooding.

Caution: If the threshold is 1 packet, you may flood the XSR and generate alarms.

For associated information on this funtionality, refer to the **`access-list`** commands on page 16-84 and page 16-86, **`show access-list log-update-threshold`** command on page 16-92, and **`logging`** command on page 3-88.

Syntax

`access-list log-update-threshold` *<number-of-packets>*

<number-of-packets>	Packets, ranging from 1 to 2,147,483,647.

Syntax of the "no" Form

Threshold logging is disabled with the *no* form of this command:

```
no access-list log-update-threshold
```

Mode

Global configuration: **XSR(config)#**

Default

Disabled

Example

The following example enables alarm logging for ACL *101* and sets the log threshold at *10000*:

```
XSR(config)#access-list 101 deny ip 15.15.15.1 0.0.0.255 16.16.16.1 0.0.0.255 log
XSR(config)#access-list log-update-threshold 10000
```

hostdos

This command enables host security protection against various DoS attacks via source IP address validation.

Note: Performing source address validation can improve security in some situations but can erroneously discard valid packets in situations where inbound and outbound paths differ and will negatively impact some routing protocols.

Syntax

hostdos {*land* | **fragmicmp** | **largeicmp** [*size*] | **checkspoof**}

land	Enables land attack protection.
fragmicmp	Enables fragmented ICMP packets protection.
largeicmp	Enables large ICMP packets protection.
size	Packet size above which protection starts, ranging from 1 to 65535.
checkspoof	Enables spoofed address checking.

Syntax of the "no" Form

The *no* form disables the specified security feature:

no hostdos {*land* | **fragmicmp** | **largeicmp** [*size*] | **checkspoof**}

Mode

Global configuration: **XSR(config)#**

Defaults

- Disabled
- Size: 1024

Example

The example below enables protection from land attack and large ICMP packets. Synflood protection will trigger for more than 7 sessions. Protection against large ICMP packets will trigger for packets larger than 2,*000* bytes.

```
XSR(config)#hostdos land
XSR(config)#hostdos largeicmp 2000
```

ip access-group

This command applies access list restrictions to an interface.

Syntax

ip access-group access *list-number* {**in** | **out**}

list-number	Number of an access list, ranging from 1 to 199.
in	Filters on inbound packets
out	Filters on outbound packets

Syntax of the “no” Form

The *no* form of this command removes the specified access group:

no ip access-group access *list-number* {**in** | **out**}

Mode

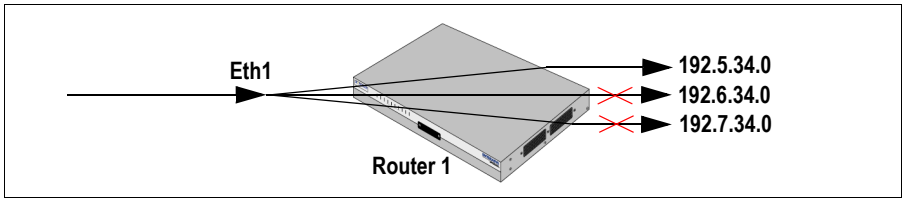
Interface configuration: **XSR(config-if<*xx*>)#**

Example

The following example, as illustrated in Figure 16-1, applies ACL *101* to all inbound packets on interface FastEthernet *1*. ACL *101* will route only packets with a destination of network *192.5.34.0*. All packets with other destinations received on FastEthernet 1 will be dropped.

```
XSR(config)#access-list 101 permit any 192.5.34.0 0.0.0.255
XSR(config)#interface FastEthernet 1
XSR(config-if<F1>)#ip access-group 1
```

Figure 16-1 IP Access-Group Example



Security Clear and Show Commands

clear hostdos-counters

This command clears all host security statistics.

Syntax

clear hostdos-counters

Mode

Privileged EXEC: **XSR#**

show access-lists

This command displays configured IP access lists. When it is issued from Global mode, it also prints a sequential *entry number* beside each ACL entry. This number can be used by the **access-list** and **no access-list** commands to specify which entries to replace, insert before, move, or delete. Since entry numbers are only useable in Global mode, (and may change when Global mode is exited) they are only displayed when in that mode.

Syntax

show access-lists [*number*]

number	ACL ID, Range: 1 to 199. If no number is specified, the entire ACL table displays.

Mode

Privileged EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following output displays when the command is issued at the Privileged EXEC mode:

```
XSR>show access-lists 101
Extended IP access list 101
   permit tcp host 18.2.32.130 any established
   permit icmp host 18.2.32.130 any
   permit tcp host 18.2.32.130 host 171.69.2.141 gt 1023
   permit tcp host 18.2.32.130 host 171.69.2.135 eq 23
   permit udp host 198.92.32.130 host 171.68.225.126 eq 45
   deny ip 11.6.0.0 0.1.255.255 224.0.0.0 15.255.255.255(
   deny ip 172.24.24.0 0.0.1.255 224.0.0.0 15.255.255.255
```

The following output displays when the command is issued at the Privileged EXEC mode:

```
XSR(config)#show access-lists

Standard IP access list 2
   1: deny   host 3.4.3.4
Extended IP access list 101
   1: permit tcp   host 2.1.2.1 any
```

show access-list log-update-threshold

This command displays ACL log information. It is processed as follows:

- A packet with a fresh source IP address on the ACL group is reported immediately. Data is cached to keep track of the occurrence happening again in the near future.
- All other arrivals of the packet with existing source IP address data on that ACL group will increment the number of packets and, after five minutes, log an alarm with the sum of packets gathered in the last five minutes. The count will reset after the alarm is logged.
- For enabled threshold data, if the count matches the threshold then the alarm is logged and the count reset. Other packets received after the threshold is met will increment the count until the next threshold is met or five minutes have elapsed.

Syntax

```
show access-list log-update-threshold
```

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following example displays a sample ACL log:

```
XSR#show access-list log-update-threshold
access-list log-update-threshold 10000
```

show hostdos

This command displays enabled host security features and their statistics.

Syntax

```
show hostdos
```

Mode

Privileged EXEC or Global configuration: **XSR#** or **XSR(config)#**

Sample Output

The following example displays a sample host security configuration with statistics:

```
XSR#show hostdos

LANd Attack (Destination IP = Source IP}
  Enabled
  10 attacks
Spoofed Address Check
  Enabled
  0 attacks
```

```
IP packet with Multicast/broadcast source address
  Always enabled
  No attacks
Syn flood attack mitigation
  Always enabled
  100 attacks
Fragmented ICMP traffic
  Enabled
  38 attacks
Large ICMP packets
  Enabled;Size 1024
  42 attacks
Ping-of-Death attack
  Always enabled
  No attack
Filter TCP traffic with Syn and Fin bits set
  Always enabled
  No attack
```

AAA Commands

The following Authentication, Authorization and Accounting (AAA) commands and command subsets validate and display information about AAA usergroups, users, and methods on the XSR:

- **aaa client**
- AAA Usergroup, User, Method amd AAA **show** commands

aaa client

This command configures sub-systems Telnet, Console, SSH (Secure Shell) and PPP to use AAA for authentication.

Syntax

aaa client {**telnet** | **console** | **ssh** | **ppp**}

telnet	Telnet sub-system.
console	Console sub-system.
ssh	SSH sub-system.
ppp	PPP sub-system.

Syntax of the No Form

The *no* form of this command resets the sub-system to use its own local AAA mechanism:

no aaa client {*telnet* | *console* | *ssh* | *ppp*}

Default

Each sub-system uses its local user database.

Mode

Global configuration: **XSR(config)#**

Examples

The following example configures the *Telnet* sub-system to use the AAA sub-system:

```
XSR(config)#aaa client telnet
```

The following example configures the *SSH* sub-system to accept AAA:

```
XSR(config)#aaa client ssh
```

AAA Usergroup Commands

aaa group

This command adds a local user group and acquires Usergroup configuration mode. Each user defined in the node must belong to one group only. The following sub-commands are available in Usergroup mode:

- **dns server** - Sets the address of DNS servers. Refer to page 16-95 for the command definition.
- **ip pool** - Links a globally defined pool of IP addresses to the user group. Refer to page 16-95 for the command definition.
- **pptp encrypt mppe** - Enables MPPE encoding on a PPTP connection. Refer to page 16-96 for the command definition.
- **privilege** - Sets the privilege level of a user. Refer to page 16-101 for the command definition.
- **wins server** - Sets the address of WINS servers. Refer to page 16-97 for the command definition.

Syntax

aaa group *group-name*

group-name	Name of the group.

Syntax of the “no” Form

The *no* form of this command deletes the group:

no aaa group *group-name*

Default

There is a default group named *DEFAULT*.

Mode

Global configuration: **XSR(config)#**

Next Mode

Usergroup configuration: **XSR(aaa-group)#**

Example

The following example adds the usergroup *headquarters:*

```
XSR(config)#aaa group headquarters
XSR(aaa-group)#
```

dns server

This command sets the address of DNS servers. These addresses are given to connecting clients during connection time.

Syntax

dns server [primary | secondary] *ip-address*

primary	Specifies primary DNS server.
secondary	Specifies secondary DNS server.
ip-address	Specifies IP address of the DNS server.

Syntax of the "no" Form

The *no* form of this command removes the configured server:

no dns server [**primary** | **secondary**] *ip-address*

Mode

Usergroup configuration: **XSR(aaa-group)#**

Example

The following example sets the primary DNS server IP address:

```
XSR(config)#aaa group headquarters
XSR(aaa-group)#dns server primary 192.168.57.9
```

ip pool

This command links a globally defined pool of IP addresses to the group of users. IP pool is defined globally by using the **ip local pool** command. If an IP pool is not linked to the group of users, each user must have an IP address configured or the connection will fail.

Note: If an **aaa user** is configured to use a static IP address which belongs to a local IP pool, you *must* exclude that address from the local pool.

Syntax

ip pool *pool-name*

pool-name	Name of the pool to be linked to the group of users. The pool-name is defined by the **ip local pool** command.

Syntax of the "no" Form

The *no* form unlinks a pool of addresses from a group of users:

no ip pool *pool-name*

Mode

Usergroup configuration: **XSR(aaa-group)#**

Example

The following example adds the IP pool *denver*:

```
XSR(config)#aaa group headquarters
XSR(aaa-group)#ip pool denver
```

pptp encrypt mppe

This command enables Microsoft Point-to-Point Encryption (MPPE) on a PPTP connection. The command must be added to the interface that will carry PPTP-MPPE traffic. All Windows clients using MPPE require MS-CHAP.

Note: All configurable MPPE options must be *identical* on both tunnel endpoints.

Syntax

pptp encrypt mppe {*auto* | *40* | *128*}

auto	Offers 40- and 128-bit encryption strength if available.
40	Only 40-bit encryption allowed.
128	Only 128-bit encryption allowed.

Syntax of the "no" Form

The *no* form of this command disables MPPE encryption:

no pptp encrypt mppe

Default

128-bit encryption

Mode

Usergroup configuration: **XSR(aaa-group)#**

Example

The following example enables MPPE with *auto* encryption:

```
XSR(config)#aaa group
XSR(aaa-group)#pptp encrypt mppe auto
```

wins server

This command sets the WINS server address which is given to connecting clients during connection time.

Syntax

wins server [**primary** | **secondary**] *ip-address*

replace	Specifies the primary WINS server.
secondary	Specifies the secondary WINS server.
ip-address	Specifies the IP address of the WINS server.

Syntax of the "no" Form

The *no* form of this command removes the configured server:

no wins server [**primary** | **secondary**] *ip-address*

Mode

Usergroup configuration: **XSR(aaa-group)#**

Example

The following example sets the secondary WINS server IP address:

```
XSR(config)#aaa group headquarters
XSR(aaa-group)#wins server secondary 192.168.57.9
```

AAA User Commands

aaa user

This command creates a new user profile in the local user database. During authentication, user-provided credentials are matched against the user's profile in the group. If you do not later associate this new user with a group, it will be added to the DEFAULT AAA group.

Note: If an **aaa user** is configured to use a static IP address which belongs to a local IP pool, you must exclude that address from the local pool.

The following sub-commands can be configured in AAA User mode:

- **group -** Specifies the group the user belongs to. Refer to page 16-98 for the command definition.
- **ip address -** Specifies the IP Address assigned to the remote user. Refer to page 16-99 for the command definition.
- **password -** Sets a user's password. Refer to page 16-99 for the command definition.
- **policy** - Configures the user's authorized list of services. Refer to page 16-100 for the command definition.
- **privilege** - Sets the privilege level of a user. Refer to page 16-101 for the command definition.

Syntax

aaa user *user-name*

user-name	Name of new user in the group; it is employed during login.

Syntax of the "no" Form

The *no* form of this command deletes the *user* profile:

no aaa user *user-name*

Mode

Global configuration: **XSR(config)#**

Next Mode

Username configuration: **XSR(aaa-user)#**

Example

The following example adds the user *ernest* to the DEFAULT usergroup:

```
XSR(config)aaa user ernest
XSR(aaa-user)#
```

group

This command specifies the group the user belongs to.

Syntax

group *group-name*

group-name	Name identifying the group a user belongs.

Syntax of the "no" Form

The *no* form of this command resets a user to the *DEFAULT* group:

no group

Default

User belongs to the DEFAULT group.

Mode

Username configuration: **XSR(aaa-user)#**

Example

The following example adds the group *run_pamplona* that the previously created user belongs to:

```
XSR(config)#aaa user ernest
XSR(aaa-user)#group run_pamplona
```

ip address

This command specifies the IP address to be assigned to the remote user. If an IP address is not specified, it is taken from the pool associated with the user's group. If an IP address is specified at the user level, it is used instead of taking a new address from the pool.

Syntax

ip address *ip-address*

ip-address	IP address to be assigned to the remote client.

Syntax of the "no" Form

The *no* form of this command removes the IP address from a user profile:

no ip address

Default

IP address is not assigned to the user.

Mode

Username configuration: **XSR(aaa-user)#**

Example

This example sets an IP address that will be assignd to remote user *ted*:

```
XSR(config)#aaa user ted
XSR(aaa-user)#ip address 192.168.57.9 255.255.255.0
```

password

This command specifies a user's password.

Syntax

password *password*

password	Password to be assigned to the user.

Syntax of the "no" Form

The *no* form of this command removes the password from a user profile:

no password *password*

Mode

Username configuration: **XSR(aaa-user)#**

Example

The following example sets the password *williams* for user *ted*:

```
XSR(config)#aaa user ted
XSR(aaa-user)#password williams
```

policy

This command configures the user's policy or authorized list of services, and it overrides the policy specified by the user's group. It is available in both AAA User and AAA Group configuration modes.

Up to four keywords can be specified in the command statement.

Syntax

policy {**vpn** | **telnet** | **console** | **firewall** | **ssh** | **ppp**} [**vpn** | **telnet** | **firewall** | **ssh** | **ppp** ...}

vpn	Sub-system keyword for VPN policy.
telnet	Sub-system keyword for Telnet policy.
console	Sub-system for Console policy.
firewall	Sub-system keyword for Firewall policy.
ssh	Sub-system keyword for Secure Shell (SSH) policy.

Note: A sub-system keyword can be stated no more than once in the command.

Syntax of the No Form

The *no* form of this command disables the earlier configured policy:

no policy {**vpn** | **telnet** | **console** | **firewall** | **ssh** | **ppp**}

Mode

AAA User/Group configuration: **XSR(aaa-user)#** or **XSR(aaa-group)#**

Example

The following example provides user access to VPN, Telnet, Console and Secure Shell (SSH), and then removes SSH from the user's policy:

```
XSR(aaa-user)#policy vpn telnet console ssh
XSR(aaa-user)#no policy ssh
```

privilege

This command configures the privilege level of a user. It is available from both AAA User and AAA Group configuration modes. Compare this command with the Interface mode **privilege** command on page 111.

Syntax

privilege *level* (0-15)

level	Specifies the privilege level (0-15) associated with this user.

Syntax of the No Form

Use the *no* form of this command to restore the privilege level default:

no privilege

Default

0

Mode

AAA User/Group configuration: **XSR(aaa-user)#** or **XSR(aaa-group)#**

Example

The following example specifies a privilege level of *15* for user *kramer*:

```
XSR(config)#aaa user kramer
XSR(aaa-user)#privilege 15
```

AAA Method Commands

aaa method

This command is executed at the Global Mode.

This command configures the AAA method (plug-in) to be used. The following sub-commands are available in AAA Method mode:

- **acct-port -** Sets the UDP port for accounting requests. Refer to page 16-103 for the command definition.
- **address -** Specifies the RADIUS server address with either a host name or IP address. Refer to page 16-103 for the command definition.
- **attempts -** Sets the number of consecutive login attempts that must fail before the RADIUS method's backup method is used. Refer to page 16-104 for the command definition.
- **auth-port -** Specifies the UDP port for authentication requests. Refer to page 16-104 for the command definition.
- **backup** - Specifies a name for a backup RADIUS method name. Refer to page 16-105 for the command definition.

- **client** - Configures the default AAA method (plug-in) for each client service. Refer to page 16-106 for the command definition.
- **enable -** Enables the current AAA server for RADIUS. Refer to page 16-106 for the command definition.
- **group -** Specifies the name of an existing group. Refer to page 16-107 for the command definition.
- **hash enable -** Enables the hash algorithm used for RADIUS. Refer to page 16-108 for the command definition.
- **key -** Sets the authentication and encryption key used between the XSR and the server daemon running on a RADIUS server. Refer to page 16-108 for the command definition.
- **qtimeout -** Specifies the queue timeout. Refer to page 16-109 for the command definition.
- **retransmit -** Specifies the number of AAA RADIUS server requests sent to a server. Refer to page 16-109 for the command definition.
- **timeout -** Sets the interval the XSR waits for the AAA RADIUS server to reply before retransmitting. Refer to page 16-110 for the command definition.

Syntax

aaa method {**local** | **radius** | **pki**} *method-name* [*default*]

local	Local AAA method.
radius	RADIUS method. You must set a RADIUS server type.
pki	PKI method.
method-name	Designation of the AAA method (plug-in).
default	If the keyword is set, the method is DEFAULT, unless overridden on a per-service basis by the **client** sub-command.

Syntax of the "no" Form

Use the *no* form to delete the AAA method and restore the default:

no aaa method {**local** | **radius** | **pki**} *method-name*

Default

If the default is not specified, the local method is the default for AAA service and subsystems lacking their own default.

Mode

Global configuration: **XSR(config)#**

Next Mode

AAA Method configuration: **XSR(aaa-method-*xx*)#**

Example

This example sets RADIUS method *sbr* as the default for AAA service:

```
XSR(config)#aaa method radius sbr default
```

acct-port

This command specifies the UDP port for accounting requests and uses the RADIUS method only.

Note: If the port number is 0, the host will not be used for accounting.

Syntax

acct-port *port-number*

port-number	Port number for accounting requests, ranging from 0 to 10,000.

Syntax of the "no" Form

The *no* form of this command resets to the default port number:

no acct-port

Default

Authorization port number: 1646.

Mode

AAA Method configuration: **XSR(aaa-method-radius)#**

Example

This example uses RADIUS *SBR* to reset the UDP accounting port to *6000*:

```
XSR(config)#aaa method radius sbr default
XSR(aaa-method-radius)#auth-port 6000
```

address

This command specifies the *address* of the RADIUS server with either a host name or IP address. It is used for the RADIUS method *only*.

Syntax

address {**host-name** | **ip-address**} *address*

host-name	Specifies the address with a host name.
ip-address	Specifies the IP address.
address	Address string: either a host name or IP address depending on which keyword is specified.

Syntax of the "no" Form

The *no* form of this command clear the address attribute:

no address {**host-name** | **ip-address**}

Mode

AAA Method configuration: **XSR(aaa-method-radius)#**

Example

The following example sets *number9* as the RADIUS server host-name:

```
XSR(config)#aaa method radius ias default
XSR(aaa-method-radius)#address host-name number9
```

attempts

This command sets the number of consecutive login attempts that must transpire before the RADIUS method's backup method is used. It is used for the RADIUS method *only*. When a user login request fails because the server did not respond, it is a failed attempt.

Syntax

attempts [*number-of-attempts*]

number-of-attempts	Sum of tries allowed, ranging from 1 to 10.

Syntax of the “no” Form

The *no* form of this command resets to the default attempts number:

```
no attempts
```

Default

4

Mode

AAA Method configuration: **XSR(aaa-method-radius)#**

Example

This example resets the attempts value to *10* on the RADIUS *IAS* server:

```
XSR(config)#aaa method radius ias default
XSR(aaa-method-radius)#attempts 10
```

auth-port

This command specifies the UDP port for authentication requests. It is used for the RADIUS method *only*.

Note: If the port number is 0, the host will not be used for authentication.

Syntax

auth-port *port-number*

port-number	Port number for authentication requests, ranging from 0 to 10,000.

Syntax of the "no" Form

The *no* form of this command resets to the default port number - 1645:

no auth-port

Default

The default authorization port number is 1645.

Mode

AAA Method configuration: **XSR(aaa-method-radius)#**

Example

The following example resets the UDP authentication port to *5000*:

```
XSR(config)#aaa method radius sbr default
XSR(aaa-method-radius)#auth-port 5000
```

backup

This command creates a name for a backup RADIUS server. The RADIUS backup method does not permit loops. That is, method 1 can have a backup method 2 but its backup method 3 cannot back up method 1. Be aware that when the primary RADIUS server fails and AAA switches to the backup, use of the primary server will not automatically be restored when it comes back on line. You must manually restart the primary server with the **aaa method radius** command.

Syntax

backup *name*

name	Designation of the backup RADIUS server.

Syntax of the "no" Form

The *no* form of this command deletes the backup RADIUS server:

no backup *name*

Mode

AAA Method configuration: **XSR(aaa-method-radius)#**

Example

The following example specifies *Radius2* as the backup server name:

```
XSR(config)#aaa method radius sbr default
XSR(aaa-method-radius)#backup Radius2
```

client

This command configures the default AAA method (plug-in) for each client service. If a client service is not registered by this command, requests from that service will fall through to the overall default method.

For example, if the authentication mode has not been set for Telnet using **aaa client telnet**, then the default AAA method set for Telnet users via the **client** command will be ignored. Telnet users will be authenticated by Telnet's AAA scheme using its own user database.

Note: You can specify a username as *username@method*, allowing that user to explicitly specify which AAA method to use for that login attempt.

Syntax

client {**vpn** | **telnet** | **firewall** | **console** | **ssh** | **ppp**}

Note: PPP uses AAA only when acting as the *authenticator* (that is, when validating the peer). PPP's client-side functionality is authenticated by the peer when acting as the *authenicatee*.

Syntax of the No Form

The *no* form of this command removes the default method for the associated client service:

no client {**vpn** | **telnet** | **firewall** | **console** | **ssh** | **ppp**}

Mode

AAA Method configuration: **XSR(aaa-method-*xx*)#**

Default

VPN access is enabled, all other access types are disabled.

Example

This example configures RADIUS method *sbr* as the default method for the client-service *Telnet*:

```
XSR(config)#aaa method radius sbr
XSR(config-aaa-rad)#client telnet
```

enable

This command enables the current AAA server for RADIUS only.

Syntax

enable

Syntax of the "no" Form

The *no* form of this command disables the current AAA server service:

no enable

Default

Enabled

Mode

AAA Method configuration: **XSR(aaa-method-radius)#**

Example

The following example enables the RADIUS server:

```
XSR(config)#aaa method radius sbr default
XSR(aaa-method-radius)#enable
```

group

This command specifies the *group* added earlier using the **aaa group** command. This command is available for all AAA methods (local, RADIUS and PKI). The group will be used when a group name is *not* returned in the RADIUS response.

Syntax

group *group-name*

group-name	The name of a valid (existing) group.

Syntax of the “no” Form

The *no* form of this command resets to the default group - *DEFAULT*:

```
no group
```

Default

DEFAULT

Mode

AAA Method configuration: **XSR(aaa-method-*xx*)#**

Example

The following example sets the group *redsox* as the default group:

```
XSR(config)#aaa group redsox
XSR(config)#aaa method local default
XSR(aaa-method-local)#group redsox
```

hash enable

This command enables the hash for the plugin and is used for the RADIUS method *only*. The sub-command may be a plugin-type dependent command.

Syntax

hash enable

Syntax of the "no" Form

The *no* form of this command disables hashing:

no hash enable

Default

Disabled

Mode

AAA Method configuration: **XSR(aaa-method-radius)#**

Example

The following example enables the RADIUS hash:

```
XSR(config)#aaa method radius sbr default
XSR(aaa-method-radius)#hash enable
```

key

This command specifies the authentication and encryption key used between the XSR and the server daemon running on this RADIUS server. The sub-command may be a plugin-type dependent command. It is used for the RADIUS method only.

Syntax

key *key-string*

key-string	Sets the authentication and encryption key for all RADIUS communications between the XSR and RADIUS server. This key must match the encryption used on the RADIUS daemon. All leading spaces are ignored, but spaces within and at the end of the key are used.

Syntax of the "no" Form

The *no* form of this command clears the key attribute:

no key

Mode

AAA Method configuration: **XSR(aaa-method-radius)#**

Example

The following example resets the RADIUS key value to *1234qwerty*:

```
XSR(config)#aaa method radius default
XSR(aaa-method-radius)#key 1234qwerty
```

qtimeout

This command specifies the interval a timeout request is allowed to sit unprocessed on AAA's internal queue before it is discarded.

Syntax

qtimeout *seconds*

seconds	Timeout value ranging from 0 to 5000 seconds.

Syntax of the “no” Form

The *no* form of this command resets to the default value:

```
no qtimeout
```

Default

30 seconds

Mode

AAA Method configuration: **XSR(aaa-method-*xx*)#**

Example

The following example sets the qtimeout to *3,600* seconds:

```
XSR(aaa-method-local)#qtimeout 3600
```

retransmit

This command specifies the number of times an AAA RADIUS server request is re-sent to a server if that server is not responding or responding slowly. It is used for RADIUS (1-5) only.

Syntax

retransmit [*retries*]

retries	Retransmit value ranging from 1 to 5.

Syntax of the “no” Form

The *no* form of this command resets the value to the default:

```
no retransmit
```

Default

3

Mode

AAA Method configuration: **XSR(aaa-method-*xx*)#**

Example

The following example lengthens the retransmit value to *5*:

```
XSR(config)#aaa method radius default
XSR(aaa-method-radius)#retransmit 5
```

timeout

This command specifies the interval, in seconds, that the XSR waits for the AAA RADIUS server to reply before retransmitting. It is used for the RADIUS method *only.*

Syntax

timeout *seconds*

seconds	Timeout value ranging from 1 to 30 seconds.

Syntax of the “no” Form

The *no* form of this command resets to the default value:

```
no timeout
```

Default

5 seconds

Mode

AAA Method configuration: **XSR(aaa-method-radius)#**

Example

The following example resets the RADIUS AAA timeout to *25* seconds:

```
XSR(aaa-method-radius)#timeout 25
```

AAA Per-Interface Commands

aaa-method

This command is executed at the Interface Mode.

This command specifies the name of the AAA method you will use for authentication requests originating from this interface. With this command, you can process authentication requests originating from different interfaces by different methods.

The command is governed by the following rules:

- If an interface has no method specified or the specified method does not exist, standard AAA method selection applies.
- The **@<method> username** syntax overrides the interface's method.
- IKE is not affected because it always employs the PKI method.
- The interface-specific method will override the service type's default method (assigned via the **client** sub-command in AAA method configuration mode) and the AAA service's default method.

Syntax

aaa method *method-name*

method-name	Designation of the AAA method (plug-in).

Syntax of the "no" Form

The *no* form of this command de-selects this method:

no aaa method

Mode

Interface configuration: **XSR(config-if<xx>)#**

Example

This example sets the PPP method for AAA service on *FastEthernet interface 2*:

```
XSR(config-if<F2>)#aaa method PPP
```

aaa privilege

This command associates the specified *interface* with a maximum privilege level available for AAA logins. Be aware that you can assign a user's privilege level based on AAA user/group information, unless it exceeds the level assigned to an interface via this command. Compare this command with the AAA Use and Group mode **privilege** command on page 101.

Syntax

aaa privilege *level*

level	Maximum privilege setting, ranging from 0 (lowest) to 15.

Syntax of the “no” Form

The *no* form of this command removes the user/group/interface restriction:

no aaa privilege

Mode

Interface configuration: **XSR(config-if<xx>#**

Default

Privilege level: 15'

Example

This example resets the privilege level to *10* on *GigabitEthernet interface 2*:

XSR(config-if<G2>)#aaa privilege 10

AAA Debug and Show Commands

debug aaa

This command activates/deactivates the output of AAA debugging data, which is classified by Authentication, Accounting and Authorization categories.

The command’s output will be sent to the terminal that most recently requested debug information. Also, if multiple AAA debug messages are activated, all debug data will be sent to the terminal from which it was most recently activated.

Syntax

debug aaa {*accounting* | *authentication* | *authorization*}

accounting	Accounting debug data displayed.
authentication	Authentication debug data displayed.
authorization	Authorization debug data displayed.

Syntax of the “no” Form

The *no* form of this command resets to the default value:

no debug aaa {*accounting* | *authentication* | *authorization*}

Mode

Privileged EXEC: **XSR#**

Sample Output

The debug authorization message below indicates the *Local* method was *successful* with MSCHAP:

Local::queue(test)

```
AAuthenticatePlugin::queue (alg == 0xf)
groupplugin Reply: Pool   = authpool
IRMauthorizeMsg::clientLogon [test]
```

The following is a debug authentication message showing the *Local* method *failed* with *MSCHAP*:

```
Local::queue(test)
AAuthenticatePlugin::queue (alg == 0xf)
(Local) Failed mschap authentication
(Local) do_ms_chap: Invalid user name or password
Method [Local]: Error for user [test] on [Authenticate]
```

show aaa group

This command displays properties of the AAA *group*.

Syntax

show aaa group *group-name*

group-name	Name of the group to be displayed. If not specified, all groups are displayed.

Default

If a group-name is not specified, all groups are displayed including the *DEFAULT* group.

Mode

Privileged EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following output is displayed by the command:

```
XSR#show aaa group

AAA Group Stats:

 Group Name: sales
 Group Comment: Toledo Branch Office
 IP Address is: 0.0.0.0
 IP Mask is: 0.0.0.0
 Primary DNS server is: 2.3.2.3
 Secondary DNS server is: 2.3.2.4
 Primary WINS server is: 3.3.2.3
 Secondary WINS server is: 3.3.2.4
 IP pool for the group is:
 PPTP encryption is 128 bit
Access Policy is: VPN
 Privilege Level is: 15

Group Name: DEFAULT
 Group Comment:
```

```
IP Address is: 0.0.0.0
IP Mask is: 0.0.0.0
Primary DNS server is: 0.0.0.0
Secondary DNS server is: 0.0.0.0
Primary WINS server is: 0.0.0.0
Secondary WINS server is: 0.0.0.0
IP pool for the group is:
PPTP encryption is 128 bit
Access Policy is: firewall
Privilege Level is: 0
```

show aaa user

This command displays *user* properties including the group to whom the user belongs and its IP address.

Syntax

show aaa user [*user-name*]

user-name	Name of the user to be displayed.

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following output is displayed by the command:

```
XSR#show aaa user

AAA User Stats:

User Name: larryj
Group Name: documentation
IP Address: 192.168.57.9
Mask: 255.255.255.0
Access Policy: SSH
Privilege Level: 15
```

show aaa method

This command displays configured plugins and their parameters.

Syntax

show AAA Method [*method-name*]

method-name	Name of the AAA method (plugin name).

Default

If the *method-name* is not set, all methods and method attributes display.

Mode

EXEC or Global configuration: **XSR>** or **XSR(config)#**

Sample Output

The following output is displayed by entering **show aaa method**:

```
XSR#show aaa method

AAA Method Stats:

Method Type: PKI
Default group name is: DEFAULT
Queue timeout is: 0
Registered Clients: VPN

Method Type: Local (Default Method)
Default group name is: acme
Queue timeout is: 5000
Registered Clients: VPN

Method Type: Radius, Method Name: def
This method is currently enabled
Backup Radius server name is: RADbackup
Default group name is: DEFAULT
IP Address is: 0.0.0.0
Hash is currently: enabled
Authentication and encryption key is: 3edue8jmdi
The UDP port for Authentication is: 1645
The UDP port for Accounting is: 1646
Maximum number of login attempts is: 4
Maximum number of retransmission tries is: 3
Attempt Timeout is: 10
Queue timeout is: 0
Registered Clients: Firewall
```

Firewall Feature Set Commands

ip firewall auth

This command defines the object which handles configuration for firewall authentication.

Syntax

ip firewall auth {**timeout** *<60-1800>* | **port** *<1024-65535>*}

timeout *#*	Idle timeout for authentication cache entry, ranging from 60 to 1800 seconds.

port #	TCP port on which the firewall authenticator will listen. Range: 1024 to 65535.

Syntax of the "no" Form

The *no* form sets either the *timeout* or *Auth port* to its default value:

no ip firewall auth {**timeout** # | **port** #}

Defaults

- Timeout: 1800 seconds
- Authentication port: 3000

Mode

Global configuration: **XSR(config)#**

Example

The following example resets the ICMP idle timeout:

```
XSR(config)#ip firewall icmp timeout 3000
```

ip firewall disable/enable

When issued in Global mode, this command is a "master switch" which activates or deactivates the firewall system-wide. You can also use this command as a "local switch" in Interface configuration mode, enabling or disabling the firewall on a per interface basis. The command behaves separately and interactively at Global and Interface modes as follows:

- The system-level firewall is *disabled* by default.
- The interface-level firewall is *enabled* by default unless explicitly disabled.
- If the firewall is enabled, packet inspection will occur on all interfaces that have the firewall enabled at the interface level.
- A particular interface may be enabled but subsequently disabling the firewall globally *overrides* all enabled interfaces.
- If you enable the firewall globally, *all* interfaces will be enabled until you subsequently disable a particular interface.
- **Enable** displays in **running-config**, but not **disable.**
- Even if you have not configured the firewall, entering **ip firewall enable** will turn on packet inspection.

Note: TCP traffic (e.g., Telnet) passed first through a firewall-disabled interface destined to a firewall-enabled will be dropped regardless of policy.

Syntax

ip firewall {**disable** | **enable**}

Default

Disabled globally

Mode

Global or Interface configuration: **`XSR(config)#`** or **`XSR(config-if<xx>)#`**

Example

The following example enables the firewall globally:

```
XSR(config)#ip firewall enable
```

ip firewall filter

This command defines the filter object for non-TCP and UDP traffic, for which no stateful inspection is required. By default, all non-TCP and UDP traffic is dropped by the firewall. To allow certain IP protocols to pass through the firewall, a filter object must be configured.

Filtering is performed on the protocol ID and source and destination addresses which are network objects. Protocols can be specified by number or name. If a name is used, it should match that specified by the Internet Assigned Numbers Authority (IANA). Refer to:

http://www.iana.org/assignments/protocol-numbers

A name for any firewall object must use these alpha-numeric characters *only*: **A** - **Z** (upper or lower case), **0** - **9**, **–** (dash), or **_** (underscore). Also, all firewall object names including pre-defined objects such as ANY_EXTERNAL and user-defined object names are case-sensitive.

Note: Logging for the filter is performed on a *per packet* basis.

Syntax

`ip firewall filter` *`filter_name src_net_name dst_net_name`* {*`protocol-id prot-number | protocol-name prot-name`*} [*`type number`*] [*`allow-log`*] **`bidirectional`**

`filter_name`	Name of filter object, not to exceed 16 characters.
`src_net_name`	Name of any source network object. Limit: 16 characters.
`dst_net_name`	Name of destination network object. Limit: 16 characters.
`protocol-id`	Protocol specified by decimal value.
`protocol-name`	Protocol specified by name, not to exceed 16 characters.
`type` *`number`*	If the protocol is ICMP, you can filter specific types only.
`bidirectional`	Policy applies in both directions. That is, for a session initiated at the source as well as the destination.
`allow-log`	All matching packets are logged.

Syntax of the “no” Form

The *no* form of this command disables the specified filter:

```
no ip firewall filter filter_name
```

Defaults

Deny all

Mode

Global configuration: **XSR(config)#**

Example

The following example permits any remote host to run a PPTP tunnel to a server on the internal network:

```
XSR(config)#ip firewall network pptp-server 120.21.1.18/32 internal
XSR(config)#ip fire filter allow--gre ANY_EXTERNAL pptp-server 47 protocol-id
XSR(config)#ip firewall filter allow--gre pptp-server ANY_EXTERNAL protocol-id 47
```

ip firewall icmp timeout

This command defines the object which handles all configuration for ICMP packet inspection.

Syntax

ip firewall icmp timeout <*seconds*>

seconds	Idle timeout for ICMP sessions, ranging from 60 to 86400 seconds.

Syntax of the “no” Form

The *no* form of this command sets the timeout to the default value:

```
no ip firewall icmp timeout
```

Default

Timeout: 60 seconds

Mode

Global configuration: **XSR(config)#**

Example

The following example resets the ICMP idle timeout interval:

```
XSR(config)#ip firewall icmp timeout 300
```

ip firewall java and ip firewall activex

This command defines the object that allows or denies HTML pages with embedded Java or ActiveX applets from particular or all IP addresses. A name for any firewall object must use these alpha-numeric characters *only*: **A** - **Z** (upper or lower case), **0** - **9**, **-** (dash), or **_** (underscore). Also, all firewall object names are case-sensitive.

Syntax

ip firewall java {**all, none, selected** *network_name*}
ip firewall activex {**all, none, selected** *network_name*}

all	Permit HTML pages with Java from all IP addresses.
none	Deny HTML pages with Java from any IP address.
selected	Permit HTML pages with Java from selected IP addresses.
network_name	Any internal or external network or network-group object.

Syntax of the "no" Form

The *no* form of this command disables Java or ActiveX:

no ip firewall java/activex {**all, none, selected** *network_name*}

Default

Deny all HTML pages with Java and ActiveX applets

Mode

Global configuration: **XSR(config)#**

Example

The following example configures *corporate-network* as a network group object listing all reachable networks, excluding any ActiveX applets, at corporate headquarters:

```
XSR(config)#ip firewall java selected corporate-network
XSR(config)#ip firewall activex none
```

ip firewall load

This command loads current firewall settings into the router's inspection engine. The current configuration comprises all CLI commands that have been entered since the last load. Executing this command clears all sessions thus requiring all TCP connections be re-established.

Because the *no* version of this command is not available, in order to undo a recent firewall configuration you must execute *no* versions of commands which invoke the configuration. Optionally, you can build the configuration but not disturb the firewall engine. This is a useful tool to configure the firewall while incrementally checking its validity. Also, you can schedule a load although this option blocks any firewall configuration in the interim.

Syntax

ip firewall load delay [**trial**]{*1-7 [hh:mm]|hh:mm*}[*enable |disable*]

trial	Builds configuration but does not load it into the firewall engine.
1-7 hh: mm:	Interval in the format days *<1-7> HH:MM* to wait until the firewall load or restart is performed. No object can be modified during this time except a trial load. Logging restarts when the load runs. The *days* value is optional and if entered, the *hours* and *minutes* values are also optional.

enable disable	Executes or terminates the firewall load.

Note: If the command is issued when a load delay is pending, the following error message displays:

```
Load: Configuration locked due to scheduled load delay
```

Syntax of the “no” Form

The *no* form of this command cancels a scheduled load and unlocks the firewall config CLI:

```
XSR(config)#no ip firewall load delay
```

Mode

Global configuration: **XSR(config)#**

Examples

The following example verifies the firewall configuration is correct:

```
XSR(config)#ip firewall load trial
```

This example schedules a load in *five days, three hours* and *20 minutes*:

```
XSR(config)#ip firewall load delay 5 03:20
```

After the load is performed, the following message will display:

```
XSR(config)#<186>Mar 17 22:30:22 10.10.10.20 FW: Firewall Shutdown and Restarted
<186>Mar 17 22:30:22 10.10.10.20 FW: Firewall: The Firewall has just executed a
delayed load command successfully
```

ip firewall logging

This command defines logging object parameters that apply to the firewall log operation. Logging is cumulative. For example, by selecting Level 3, the firewall will generate all messages from Levels 3 to 0. If you set logging to Level 0, the number of messages will be minimal.

Levels 0 to 3 are designated for *attacks*, *denies* and other system-related logs such as memory failures. Levels 4 to 7 are designated for *permits*, *warnings* and other informational logs. There are very few debug level logs so in order to see permits a setting of 5 or 6 is sufficient.

Syntax

ip firewall logging event-threshold *0-7*

event-threshold	Events of severity equal to or lesser than the specified value log as follows: • Level 0: **Emergency** • Level 1: **Alert** • Level 2: **Critical** - alarms such as *failure to allocate memory during initializiation* are logged if system logging is enabled and firewall logging is set to level 2 or higher • Level 3: **Error** - *abnormal* and *deny* alarms are logged if system logging is set at MEDIUM or HIGH and firewall logging is level 5 or higher • Level 4: **Warning** - *normal* and *permit* alarms are logged if system logging is set at LOW and firewall logging is level 4 or higher • Level 5: **Notice** • Level 6: **Information** • Level 7: **Debug**

Syntax of the "no" Form

The *no* form of this command sets firewall logging to the default value:

no ip firewall logging event-threshold

Default

Level 3 - All denies and series faults are logged

Mode

Global configuration: **XSR(config)#**

Example

This example sets firewall logging for all messages Notice level:

```
XSR(config)#ip firewall logging 5
```

ip firewall network

This command defines a network object specifying a network or host IP address or address group (base and subnet mask or start and end IP address) that is tagged as internal or external. Naming a location is helpful in using this object for rules indicating any internal/external network.

Network objects are referenced by the name within the policy and network group objects. Define network objects for internal hosts and networks. A name for any firewall object must use these alpha-numeric characters *only*: **A** - **Z** (upper or lower case), **0** - **9**, **-** (dash), or **_** (underscore).

Also, all firewall object names including pre-defined objects such as ANY_EXTERNAL and user-defined object names are case-sensitive.

Notes: A DMZ is considered an *internal* network.

Use care when you have a configuration with internal and external addresses that overlap and exist off the same physical interface. In this case, the XSR may not be able to identify an address in the overlap range as being internal or external. If this is so, packets may not match policies as expected.

Once you specify a network name you cannot switch internal/external settings. To switch settings you must delete the network and add it again.

Syntax

ip firewall network *name* {*A.B.C.D* **mask** *A.B.C.D* | *A.B.C.D A.B.C.D*}{**internal** | **external**}

name	Name of the network object, not to exceed 16 characters. Match this with **policy** source/destination name *exactly*.
A.B.C.D A.B.C.D	Start and end addresses.
A.B.C.D **mask** *A.B.C.D*	Base address and mask in dotted decimal format.
internal or **external**	Address qualifier.

Syntax of the "no" Form

The *no* form of this command disables the firewall network object:

no ip firewall network *name*

Syntax

Global configuration: **XSR(config)#**

Example

This example defines internal and external IP addresses for the network objects *sales* and *remote-access*. Note how the internal and external tags have meaning in the way the network objects are used in a policy.

```
XSR(config)#ip firewall network sales 192.168.100.0 mask 255.255.255.0 internal
XSR(config)#ip firewall network remote-access 10.1.1.0 mask 255.255.255.0 external
```

ip firewall network-group

This command comprises a set of network objects, serving the same function as a network object. Intrinsic values ANY_INTERNAL (all internal network objects defined) and ANY_EXTERNAL (all external network objects defined) are a convenient option to define a set of network objects.

Membership in these sets is unlimited.

A name for any firewall object must use these alpha-numeric characters *only*: **A** - **Z** (upper or lower case), **0** - **9**, **–** (dash), or **_** (underscore). Also, all firewall object names including pre-defined

objects such as ANY_EXTERNAL and user-defined object names are case-sensitive. Refer to the **ip firewall policy** command for applicable policy and gating rule limits.

Syntax

ip firewall network-group *name name1 ... name10*

name	Network group object name. Limit: 16 characters.
name1 to name10	Name of the network or network-group objects.

Syntax of the "no" Form

The *no* form of this command disables the network group:

no ip firewall network-group *name*

Mode

Global configuration: **XSR(config)#**

Example

The following example defines network objects *sales* and *remote-access* and adds them to the network groups *private-net* and *sales remote-access*:

```
XSR(config)#ip firewall network sales 192.168.100.0 ma 255.255.255.0 i
XSR(config)#ip fi network remote-access 10.1.1.0 m 255.255.255.0 i
XSR(config)#ip firewall network-group private-net sales remote-access
```

ip firewall policy

This command configures a firewall policy comprised of policy objects. Each object/rule is tagged with a name which places the policies in order using a before and after keyword. This permits you to enter policies in an order different than which they will be applied.

The XSR firewall enforces a *deny all* policy by default. So, unless there is a policy object configured to allow traffic in a particular direction, packets will not pass through the firewall. This eliminates the need to define catch-all reject policies in each direction.

Policies apply to traffic directed at the router, as well. So, policy objects must be defined to allow management traffic into the router. Be aware that the console port is always available for management purposes.

A name for any firewall object must use these alpha-numeric characters *only*: **A** - **Z** (upper or lower case), **0** - **9**, **–** (dash), or **_** (underscore). Also, all firewall object names including pre-defined objects such as ANY_EXTERNAL and user-defined object names are case-sensitive.

Notes: Citing a policy's intent in the name is useful if its function is not apparent from the definition.

Internal XSR gating rules, which order traffic filtering, are stored in a temporary file in Flash. Because there is one gating rule for each network source/destination expansion, a potentially enormous number of gating rules can be generated by just a single firewall policy. For example, when a large network that has an ANY_INTERNAL group with 200 network addresses is used as the source address, and another group of 10 network addresses is used as the destination address, 2000 gating rules are defined for the policy. Accordingly, a limit is applied to their total, depending on the amount of installed RAM.

Syntax

ip firewall policy *policy_name src_net_name dst_net_name serv_name* {**allow** | **allow-log** | **allow-auth** *group_name* | **reject** | **log** | **url-b** | **url-w** | **cls** *name ... name*}[**before** *policy_name* | **after** *policy_name* | **first**] [**bidirectional**]

src_net_name	Name of source network object, not to exceed 16 characters. This value must match **network** name *exactly.*
dst_net_name	Name of destination network object, not to exceed 16 characters. This value must match **network** name *exactly.*
serv_name	Name of service object, not to exceed 16 characters.
allow	Let packets pass through the firewall.
allow-log	Let packets through the firewall and log the activity.
allow-auth *group_name*	Let packets pass if the source IP address has been authenticated against the *group_name* (length not to exceed 16 characters). This value must match **network-group** name *exactly.*
reject	Drop all packets matching the policy.
log	Drop all matching packets and log the activity.
url-b | **url-w**	Filters HTTP traffic (TCP connection with a destination port of 80 or 8080) using the black (*url-b*) URL list. Filters http traffic using the white (*url-w*) URL list. HTTP access to URLs matching an entry in the white URL list are allowed, non-matching URLs are blocked.
cls *name*	Let packets pass through the firewall if the application message type matches one of the 10 type names. Names must not exceed 16 characters.
before or **after** *policy_name*	Place policy before or after the policy cited by *policy_name* (which must already have been set). If not specified, the object will be the last listed.
first	Place policy first.
bidirectional	Policy applies in both directions. That is, for a session initiated at the source as well as the destination.

Note: If the action is *allow-auth* the *group_name* must be specified. All users who are members of this group are allowed authenticated access. Also, be sure to match the *group_name* and AAA *group* name.

Syntax of the "no" Form

The *no* form of this command disables an earlier configured policy:

no ip firewall policy policy_name

Defaults

Deny all

Mode

Global configuration: **XSR(config)#**

Example

The following policy allows FTP access to a host. Be aware that the host's source IP address will be authenticated against the group *sales-group*.

```
XSR(config)#ip firewall network sales-host 192.168.100.2 mask 255.255.255.255
internal
XSR(config)#ip firewall policy allow-eng-ftp ANY_INTERNAL sales-host ftp allow-
auth sales-group
```

ip firewall redirectURL

This command redirects a user's HTTP access to the specified re-directURL page if that user attempts to access a URL not permitted by the white URL list. If re-directURL is not configured, the XSR generates a default blocked page.

Note: This command takes effect immediately.

Syntax

ip firewall redirectURL *redirect_url_string*

redirect_url_string	A valid URL string of up to 63 characters.

Syntax of the "no" Form

The *no* form of this command removes a previously configured redirectURL:

```
no ip firewall redirectURL
```

Mode

Global configuration: **XSR(config)#**

Example

The following example redirects a user to the specifed URL site:

```
XSR(config)#ip firewall redirecturl www.companyXYZ.com.
```

ip firewall rpc timeout

This command sets the idle session timeout on packet inspection for Remote Procedure Call (RPC) -based applications. This Application Level Gateway (ALG) supports two types of RPCs - *SUN* (used by most UNIX systems) and *Microsoft*. If the RPC-based session is idle for the specified period, it will be shut down.

Syntax

ip firewall rpc {*microsoft-rpc* | *sun-rpc*} **timeout** *number*

microsoft-rpc	ALG packet inspection for Microsoft traffic.
sun-rpc	ALG packet inspection for SUN traffic.
number	Idle session timeout, ranging from 5 to 86400 seconds.

Syntax of the "no" Form

The *no* form of this command sets the default RPC timeout value:

```
no ip firewall rpc timeout
```

Default

5 seconds

Mode

Global configuration: **XSR(config)#**

Example

The following example resets the Microsoft RPC idle timeout interval to 10 minutes:

```
XSR(config)#ip firewall rpc microsoft-rpc timeout 6000
```

ip firewall service

This command defines a service object which reflects an application, its transport protocol (TCP or UDP), protocol type and port number ranges. The XSR supports a number of pre-defined services which can be viewed with **show ip firewall user-services**. Services can be directly cited in policy objects or you can add your own service. Intrinsic services ANY_TCP and ANY_UDP are available for all TCP or UDP ports.

A service is comprised of a source and destination port range, and protocol. For flexibility, port ranges can be specified using qualifiers such as *eq*, *lt* and *gt* which are also available for configuring access lists.

A name for any firewall object must use these alpha-numeric characters *only*: **A** - **Z** (upper or lower case), **0** - **9**, **–** (dash), or **_** (underscore). Also, all firewall object names are case-sensitive.

Note: The **show ip firewall service** command displays pre-defined services.

Syntax

ip firewall service *name <source-port-range> <dest-port-range> <protocol>*
ip firewall service *name* {**eq** *<0-65535>* | **gt** *<0-65535>* | **lt** *<0-65535>* | **range** *<0-65535> <0-65535>*} {**eq** *<0-65535>* | **gt** *<0-65535>* | **lt** *<0-65535>* | **range** *<0-65535> <0-65535>*}{**tcp** | **udp**}

name	Name of the protocol, not to exceed 16 characters.
eq	Port range equals number specified.
gt	Port range is strictly greater than the number specified, and less than or equal to 65535.
lt	Port range is strictly less than the number specified.
range	Explicit port range with the start and end ranges specified: *<0-65535>*
tcp or **udp** *protocol*	Transport protocol. The *protocol* value is *case-sensitive*.

Syntax of the "no" Form

The *no* form of this command disables the selected service:

no ip firewall service *name*

Mode

Global configuration: **XSR(config)#**

Example

The following example defines the FTP service (although this is un-necessary as it is one of the pre-defined services). The source port range could be any of the un-reserved ports but the destination must be *21*.

```
XSR(config)#ip firewall service ftp gt 1023 eq 21 range 21 22 tcp
```

ip firewall service-group

This command permits the aggregation of more than one service object, providing for easier policy configuration. Up to ten service objects (and service group) can be included in a service group.

A name for any firewall object must use these alpha-numeric characters *only*: **A** - **Z** (upper or lower case), **0** - **9**, **-** (dash), or **_** (underscore). Also, all firewall object names are case-sensitive.

Syntax

ip firewall service-group *name name1 ... name10*

name	Name of the service group object, not to exceed 16 characters.
name1 to name10	Name of the service or service-group objects.

Syntax of the "no" Form

The *no* form of this command disables an earlier configured service group:

no ip firewall service-group *name*

Mode

Global configuration: **XSR(config)#**

Example

The following example configures service group *netbios* with *netbios1* and *netbios2* using ports *137* and *138*, respectively, included as service objects:

```
XSR(config)#ip firewall service netbios1 137-137 137-137 udp
XSR(config)#ip firewall service netbios2 138-138 138-138 udp
XSR(config)#ip firewall service-group netbios netbios1 netbios2
```

ip firewall tcp/udp timeout

This command resets the idle timeout interval for Firewall sessions applying TCP or UDP packet inspection. If the Firewall session is idle for the specified period, it will be shut down.

Syntax

ip firewall {*tcp* | *udp*} **timeout** <*number*>

tcp	Packet inspection for TCP traffic.
udp	Packet inspection for UDP traffic.
number	Idle timeout for TCP or UDP sessions, ranging from 60 to 86400 seconds.

Syntax of the "no" Form

The *no* form of this command sets the *default* TCP timeout value:

no ip firewall {*tcp* | *udp*} **timeout**

Default

60 seconds

Mode

Global configuration: **XSR(config)#**

Example

The following example sets the firewall session for UDP traffic to time out if idle for 10 minutes:

```
XSR(config)#ip firewall udp timeout 6000
```

ip firewall url-load-black/white-list

This command clears the specified Black URL or the White URL database then re-loads it from a specified file.

Syntax

ip firewall url-load-black-list | url-load-white-list *filter_file_name*

filter_file_name	Name of the ASCII file, containing up to 30 URL lists. The file name can be prefixed with the optional driver ID **flash:** or **cflash:**.

Syntax of the "no" Form

The *no* form of this command deletes a previously loaded URL list:

no ip firewall rpc timeout

Mode

Global configuration: **XSR(config)#**

Examples

The following examples configure valid inputs:

```
ip firewall url-load-black-list blacklist.txt
ip firewall url-load-black-list flash:blacklist.txt
ip firewall url-load-white-list cflash:whitelist.txt
```

Firewall Interface Commands

ip firewall disable

This command disables firewall operation on a particular interface discrete from its application globally. The command behaves separately and interactively at Global and Interface modes as follows:

- The system-level firewall is *disabled* by default.
- The interface-level firewall is *enabled* by default unless explicitly disabled.
- If the firewall is enabled, packet inspection will occur on all interfaces that have the firewall enabled at the interface level.
- A particular interface may be enabled but subsequently disabling the firewall globally *overrides* all enabled interfaces
- If you enable the firewall globally, *all* interfaces will be enabled until you subsequently disable a particular interface
- **`Enable`** displays in **`running-config`**, but not **`disable`**
- Even if you have not configured the firewall, entering **`ip firewall enable`** will turn on packet inspection.

Note: With the firewall enabled, source address validation (HostDoS checkspoof) is also enabled. This service can improve security in some situations but erroneously discard valid packets in situations where inbound and outbound paths differ as well as negatively impact some routing protocols.

Syntax

```
ip firewall disable
```

Syntax of the “no” Form

The *no* form of this command enables the firewall on a selected interface:

```
no ip firewall disable
```

Default

Enabled

Mode

Interface configuration: **`XSR(config-if<xx>)#`**

Example

The following example disables the firewall on *FastEthernet port 2* only:

```
XSR(config-if<F2>)#ip firewall disable
```

ip firewall ip-broadcast

This command allows incoming/outgoing IP packets through the firewall with 255.255.255.255 set as the destination address. It enables broadcast protocols such as DHCP to traverse the firewall.

Syntax

ip firewall ip-broadcast {**in** | **out** | **both**}

in or **out**	Allows packets to enter *or* exit the interface.
both	Allows packets to enter *and* exit the interface.

Syntax of the "no" Form

The *no* form of this command denies the selected broadcast packets:

no ip firewall ip-broadcast {**in** | **out** | **both**}

Default

IP broadcast packets are *not* allowed inbound and outbound.

Mode

Interface configuration: **XSR(config-if<xx>)#**

Example

The example below allows broadcast filtering on *outgoing* packets only:

```
XSR(config-if<F2>)#ip firewall ip-broadcast out
```

ip firewall ip-multicast

This command allows incoming/outgoing IP packets with a multicast destination address through the firewall. It enables multicast protocols such as RIP and OSPF to traverse the firewall.

Syntax

ip firewall ip-multicast {**in** | **out** | **both**}

in or **out**	Allows packets to enter *or* exit the interface.
both	Allows packets to enter *and* exit the interface.

Syntax of the "no" Form

The *no* form of this command denies the selected multicast packets:

```
no ip firewall ip-multicast {in | out | both}
```

Default

Multicast packets are *not* allowed inbound and outbound.

Mode

Interface configuration: **XSR(config-if<xx>)#**

Example

The following example permits multicast packets in both directions:

```
XSR(config-if<F1>)#ip firewall ip-multicast both
```

ip firewall ip-options

This command allows incoming/outgoing packets through the firewall with the following options: *loose* and *strict* source routing, *record* route, *time stamp*, *all* and *other* IP options.

Syntax

```
ip firewall ip-options {loose-source-route | strict-source-route | record-route |
time-stamp | other | all} {in | out | both}
```

loose-source-route	Requests routing that includes the specified routers. This routing path includes a sequence of IP addresses a datagram must follow to its destination but allows multiple network hops between successive addresses on the list.
strict-source-route	Specifies an exact route through the Internet. This routing path includes a sequence of IP addresses a datagram must follow, hop by hop, from its source to destination. The path between two successive addresses in the list must consist of a single physical network.
record-route	Traces a route. It allows the source to create an empty list of IP addresses and arrange for each router that router that handles a datagram to add its IP address to the list. When a datagram arrives, the destination device can extract and and process the list of addresses.
time-stamp	Records timestamps along a route. It is similar to the *record-route* option in that every router from source to destination adds its IP address, and a timestamp, to the list. The *time-stamp* notes the time and date a router handled the datagram, expressed in milliseconds since midnight, Universal Time.
other	Any IP option other than those explicitly supported by the command.
all	All IP options allowed.
in or **out**	Packets entering *or* exiting an interface.
both	Packets entering *and* exiting an interface.

Syntax of the “no” Form

The *no* form of this command disables the selected IP option:

```
no ip firewall ip-options {loose-source-route | strict-source-route | record-
route | time-stamp | other | all} {in | out | both}
```

Default

IP options are *not* allowed inbound and outbound.

Mode

Interface configuration: **XSR(config-if<xx>)#**

Example

The following example sets *loose source routing* on both incoming and outgoing packets at *F2*:

```
XSR(config-if<F2>)#ip firewall ip-op loose-source-route both
```

ip firewall sync-attack-protect

The SYNC attack monitor/blocker isolates a host that generates a flood of SYNC packets to the XSR's firewall and blocks traffic from that specific host, while allowing data packets to pass.

Syntax

ip firewall sync-attack-protect {block-host | **check-host** | **sync-queue**} **threshold** [*threshold*]

block-host	Block host when sync packet rate exceeds this value (sync packets/sec). The XSR can block up to 20 hosts at any given time. When blocked, all sync packets to and frames host are dropped, while other packets are allowed to go through. XSR automatically unblock host when the sync packet rate of the host drops to zero for 25 seconds. Threshold range is 10 - 5,000, default is 100
check-host	Starts to monitor sync packet rate of each host of a Class C subnet if the sync packet rate of the subnet exceeds this value. The XSR can monitor up to 3,000 class C subnets. Threshold range is 10 - 5,000, default is 100
sync-queue	Initiates sync attack protection when sync backlog queue exceeds this value. Range is 50 to 5,000, default is 500.
threshold	The limit in which the above parameters are enabled.

Syntax of the "no" Form

The *no* form of this command disables the function:

```
no ip firewall sync-attack-protect {block-host | check-host | sync-queue}
threshold
```

Mode

Interface configuration: **XSR(config-if<xx>)#**

Example

The following example blocks the host when the sync packets exceed 1000 packets per second:

```
XSR(config-if<F2>)#ip firewall sync-attack-protect block-host threshold 1000
```

Firewall Show Commands

show ip firewall config

Since the firewall is configured in a two-step process, the XSR provides a means to view the un-committed configuration. This command displays the firewall configuration combining existing commands with those entered recently, which permits a view of the complete firewall configuration with modifications.

If no firewall commands were executed since the last load then the running configuration will be displayed.

If this command is issued after the firewall commands were entered but before a firewall load was performed, the following text appears:

```
Uncommitted Firewall Configuration:
```

If the command is issued after a firewall load was performed, the following text appears:

```
Committed Firewall Configuration:
```

Syntax

```
show ip firewall config
```

Mode

EXEC or Privileged EXEC Mode: **XSR>** or **XSR#**

Sample Output

The following is sample output of the command:

```
Firewall configuration
Modified but not loaded: Yes

Ip firewall network dmz 220.150.2.16/28 internal
Ip firewall network private 220.150.2.32/28 internal
!
! Log only critical events
!
ip firewall system event-threshold 3
!
! Policies: between private and dmz
!
Ip firewall policy private dmz HTTP allow
Ip firewall policy dmz private HTTP allow
Ip firewall policy private dmz SMTP allow
```

```
Ip firewall policy dmz private SMTP allow
!
! Policies: between dmz and external
!
Ip firewall policy ANY_EXTERNAL dmz HTTP allow
Ip firewall policy dmz ANY_EXTERNAL HTTP allow
Ip firewall policy ANY_EXTERNAL dmz SMTP allow
Ip firewall policy dmz ANY_EXTERNAL SMTP allow
!
!  Policy: Allow any from private to the external
!
Ip firewall private ANY_EXTERNAL any allow
!
ip firewall filter private dmz 17
ip firewall filter private ANY_EXTERNAL 17
ip firewall filter ANY_EXTERNAL dmz 17
```

displays configuration objects associated with the firewall and values which are always in effect:

```
Modified firewall configuration:
ip firewall Network Dmz 220.150.2.16/28 Internal
ip firewall Network Private 220.150.2.32/28 Internal
ip firewall system event-threshold 3
ip firewall policy private dmz http allow
ip firewall policy dmz private http allow
ip firewall policy private dmz smtp allow
ip firewall policy dmz private smtp allow
ip firewall policy any_external dmz http allow
ip firewall policy dmz any_external http allow
ip firewall policy any_external dmz smtp allow
ip firewall policy dmz any_external smtp allow
ip firewall private any_external any allow
ip firewall filter private dmz 17
ip firewall filter private any_external 17
ip firewall filter any_external dmz 17

Values always in effect:
ip firewall udp timeout 3600
ip firewall icmp timeout 1200
ip firewall logging event-threshold 5
The Firewall is currently enabled
```

show ip firewall filter

This command displays all configured firewall filters.

Syntax

```
show ip firewall filter [name]
```

Mode

EXEC or Privileged EXEC Mode: **XSR>** or **XSR#**

Sample Output

The following output displays

```
Filter     Source        Destination      Protocol      ICMP      Bi/Log
Name       Network       Network          Name/Number   Type

noICMP     dmz           private          ICMP          N/A       Y/N
```

show ip firewall network

This static counter shows all network objects configured. If a network object name is specified then only that object is displayed.

Syntax

show ip firewall network [**name**]

Mode

EXEC or Privileged EXEC Mode: **XSR>** or **XSR#**

Sample Output

This output displays a network object for the *Engineering* firewall in the 192.168.100.0/24 range:

```
Name              Start Address    End Address      Internal/External
Engineering       192.168.100.1    192.168.100.254  internal
```

show ip firewall network-group

This static counter shows all network group objects. If a network group object is also specified then only that network group is displayed.

Note: Although **ANY_INTERNAL** and **ANY_EXTERNAL** objects do not display when this command is entered, entering **show ip firewall ANY_INTERNAL** or **ANY_EXTERNAL** will display the members of these intrinsic groups.

Syntax

show ip firewall network-group [*name*]

Mode

EXEC or Privileged EXEC Mode: **XSR>** or **XSR#**

Sample Output

The output below displays network objects for the *Private-network* and *Partner-networks* groups. Note that only member objects names are shown.

You can enter the **show ip firewall network** command to get address ranges of each network object.

```
Name                    Network (group) objects
Private-network         internet
                        Remote-access   10.1.0.0/16
Partner-networks        dmz
ext192                  ext253
                        ext254
int                     int40
```

show ip firewall service

This static counter displays all configured service objects. It includes three versions:

- **Show ip firewall service** - Displays all services, pre-defined and user-defined.
- **Show ip firewall user-defined** - Displays user-defined services only.
- **Show ip firewall service** *name* - Displays a specific service object identified by name.

Syntax

show ip firewall service [**user-defined** | *name*]

user-defined	Lists user-defined services only.
name	Name of a service object.

Mode

EXEC or Privileged EXEC Mode: **XSR>** or **XSR#**

Sample Output

The following output displays firewall service objects:

```
Name            Source port range   Destination port range     Protocol

ftp             1024-65535          21-21                      tcp
netbios         137-137             137-137                    udp
```

show ip firewall service-group

This static counter displays all service group objects. If the optional service group name is specified then only that service group object is displayed.

Syntax

show ip firewall service-group [*name*]

Mode

EXEC or Privileged EXEC Mode: **XSR>** or **XSR#**

Sample Output

The following output displays firewall service group data:

```
Name                    Service objects
all-my-tcp-services     my-ftp
                        my-telnet
```

show ip firewall policy

This static counter displays all policy objects in the order they will be applied. If a name is specified then only that policy object is displayed.

Syntax

show ip firewall policy [*name*]

name	Name of the policy object to display.

Mode

EXEC or Privileged EXEC Mode: **XSR>** or **XSR#**

Sample Output

The following sample output displays configured firewall policies:

```
Name      Source Network      Destination Network     Service   Action

outftp    admin               ANY_EXTERNAL            ftp       allow
outhttp   priv-network        ANY_EXTERNAL            http      allow
inftp     partner1            sales                   ftp       allow-auth mkt
```

show ip firewall sessions

This dynamic counter displays firewall data regarding TCP, UDP and ICMP sessions that have passed through the firewall since it was enabled.

Syntax

show ip firewall sessions [**tcp** | **udp** | **icmp**]

tcp	Displays only TCP sessions.
udp	Displays only UDP sessions.
icmp	Displays only ICMP sessions.

Note: Sessions do not display for IP broadcast packets.

Mode

EXEC or Privileged EXEC Mode: **XSR>** or **XSR#**

Default

If no options are specified all sessions are displayed.

Sample Output

The following sample output displays current firewall sessions:

```
XSR#show ip firewall sessions icmp

Source Address   Port   Dest. Address    Port   Protocol  Creation      Time/Date
192.168.100.100  0      192.168.1.103    0      ICMP      20:28:02      03-01-2002
192.168.100.100  0      192.168.1.20     0      ICMP      20:28:42      03-01-2002
```

show ip firewall auth

This dynamic counter displays the IP addresses that have been authenticated along with the group-name.

Syntax

```
show ip firewall auth
```

Mode

EXEC or Privileged EXEC Mode: **XSR>** or **XSR#**

Sample Output

The following sample output displays host authentication data:

```
XSR#show ip firewall auth

IP Address       Groupname        Idle Time (secs)
192.168.1.10     Sales            45
```

show ip firewall general

This dynamic counter displays firewall summary statistics.

Syntax

```
show ip firewall general
```

Mode

EXEC or Privileged EXEC Mode: **XSR>** or **XSR#**

Sample Output

The following sample output displays summary statistics:

```
Overall Firewall Status: Enabled

Protected Interfaces:
FastEthernet2

Unprotected Interfaces:
FastEthernet1

Session Information
-----------------------------------------------------------
        active     peak       blocked    last blocked at (UTC)   External Hosts
TCP     65         6531       0          N/A                     867
UDP     5          1271       0          N/A                     234
ICMP    0          0          3          08:20:12 FEB-03-2005    0
Total   0          0          3

Blocked DOS Attacks
-------------------
Land:             0
Christmas Tree:   0
Ping of Death:    0
Anti-Spoofing:    0
ICMP Flood:       0
Smurf:            0
SYN Flood:        370393
Tear Drop:        0

TCP Backlog Queue Length: 23
TCP Backlog Queue Congested: Yes

     Subnets tracked = 43,  Subnets exceeding check-host-threshold = 1,
     Total TCP Sessions = 1230268, TCP session Rate = 1509/sec

     Sync Attack     Source hosts blocked:
     192.168.50.4    192.168.50.99
     Sync Attack Victim hosts blocked:

Number of Gating Rules: 2
```

show ip firewall URLList

This command displays the configured URL filter information.

Syntax

```
show ip firewall URLList
```

Mode

EXEC or Privileged EXEC Mode: **XSR>** or **XSR#**

Example

The following is sample output from the command:

```
show ip firewall urLlist

  Black URLs from File: blacklist.txt

 1. www.cisco.com
 2. www.playboy.com
 3. readme.eml
 4. amber.cl

  White URLs from File: NOT LOADED

  Redirect URL: www.msnbc.com
```